SmarterDeals: A Context-aware Deal Recommendation System based on the SmarterContext Engine

by

Sahar Ebrahimi

BEng., Tarbiat Moallem University, 2004

A Thesis Submitted in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

© Sahar Ebrahimi, 2012

University of Victoria

All rights reserved. This thesis may not be reproduced in whole or in part, by photocopying or other means, without the permission of the author.

SmarterDeals: A Context-aware Deal Recommendation System based on the SmarterContext Engine

by

Sahar Ebrahimi

BEng., Tarbiat Moallem University, 2004

Supervisory Committee

Dr. Hausi A. Müller, Co-Supervisor

(Department of Computer Science)

Dr. Alex Thomo, Co-Supervisor

(Department of Computer Science)

Supervisory Committee

Dr. Hausi A. Müller, Co-Supervisor

(Department of Computer Science)

Dr. Alex Thomo, Co-Supervisor

(Department of Computer Science)

ABSTRACT

Daily-deal applications are popular implementations of online advertising strategies that offer products and services to users based on their personal profiles. Current implementations are effective but can frustrate users with irrelevant deals due to stale profiles. To fully exploit the value creation and revenue generation potential of these applications, deals must become smarter. This research presents SmarterDeals, a deal recommendation system that exploits users changing personal context information to deliver highly relevant offers. To improve the relevance of offers, SmarterDeals relies on collaborative filtering recommendation algorithms and SmarterContext, our adaptive context management framework. SmarterContext provides SmarterDeals with up-to-date information about users locations as well as product and service preferences gathered from their past and present web interactions and experiences. We validated our approach using a data set of 271,418 product and service category ratings and 65,411 real users. We present our results using a comparative analysis that involves other well known recommendation approaches.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank:

Dr. Hausi Müller and Dr. Alex Thomo, my supervisors, for their most valuable encouragement, guidance and support. I consider it an honour to work with them and I am very grateful to them for all of their help.

Norha Villegas, for all her support and encouragement and I would like to share the credit of my work with her. This thesis would not have been possible without her help.

My parents, for their endless love and support.

Chapter 1

Introduction

1.1 Problem Definition and Motivation

Daily-deal applications are marketing strategies widely used by businesses to advertise products and services using discount coupons. To receive daily-deal offers, users must create their personal profile by registering their personal information, and selecting relevant product or service categories from the list of available options. Using this information, daily-deal applications send offers that match a user's profile. These solutions use different communication channels such as e-mail, short message services, social networks, mobile applications, and websites.

GROUPON,[1] with approximately 51 million subscribers in 563 cities worldwide, is the most popular provider of coupons online [47]. Value creation in GROUPON's business model is based on the negotiation of attractive discounts with popular businesses, and the delivery of these discount offers to its subscribers via e-mail. Businesses share with GROUPON a percentage of the discounted price for every effective coupon, this is the basis of its revenue generation strategy. Despite the evident success of daily-deal businesses such as GROUPON, their value creation and revenue generation can

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

be considerably more effective by improving the relevance of delivered coupons to users. Users are spammed continuously with discount offers that, although generous, lack of relevance with respect to their changing preferences, needs and situations [22]. The delivery of irrelevant offers is a consequence of a lack of knowledge about users. To tackle this problem, daily-deal platforms must become smarter, that is, become context-aware.

Context can be defined as any information useful to characterize entities that affect the situation of users [2]. To exploit the value of this information, context must be gathered from the environment, processed to infer new contextual facts about entities and users even under changing context monitoring requirements, and provisioned to context-aware applications effectively [51]. Context Information is highly dynamic since the situations of users change over time. Therefore, to deliver relevant offers to users, the analysis of personal information provided by the user during the registration process is neither enough nor effective. On the one hand, product categories that were relevant to the user at registration-time may become irrelevant over time. On the other hand, demanding from users the manual registration of changes in this information is inconvenient. Dealing with the dynamic nature of context information, in a transparent way for the user, is a big challenge for businesses to deliver product and services based on the understanding of people's desires.

1.2 Research Methodology

To develop a context-aware user-centric system, we built our application based on the SmarterContext framework [52]. SmarterContext is a dynamic context management infrastructure that monitors the interactions of users with web entities to gather relevant context information. In SmarterContext the user is the

main controller of the context management process. First, SmarterContext identifies relevant context information from the interactions of the user with web entities. For example, when an online shopper adds products to her personal wish list, SmarterContext marks these product categories as context information useful to understand the user's preferences. Second, the user has full control over the privacy and security of her context information. Hence, SmarterContext shares context information only with context-aware applications authorized by the user. The information gathered by the SmarterContext framework about a particular user is stored into a persistent repository named the *personal context sphere*. SmarterContext uses this information to provide context-aware applications such as daily-deal applications with information useful to understanding users' situations and preferences.

In this thesis, we introduce SmarterDeals, our deal recommendation system for smarter commerce which is aware of changing personal context information to deliver coupons highly relevant to users. Smarter commerce is an IBM initiative aimed at revolutionizing commerce by exploiting hardware and software technologies. From the perspective of customers, its goal is to realize *efficient and user-centric shopping experiences* [28]. To improve the relevance of deal offers and thus advance towards smarter commerce, SmarterDeals relies on recommendation algorithms based on collaborative filtering and personal context information provided by SmarterContext. We demonstrate how the accuracy of recommendation algorithms can be improved considerably by taking into account personal context information. Therefore, we proposed a new recommendation algorithm based on traditional user-based collaborative filtering techniques [4], and Koren and Bell's algorithm [35, 36]. In contrast to these approaches, our algorithm uses context information from different users to improve the accuracy of daily-deal recommendations. In this case study our approach ex-

ploits the main types of context information: product or service preferences based on rankings, and location. SmarterContext provides our SmarterDeals recommendation algorithm with up-to-date information about users' product and service preferences gathered from their past and present web interactions and experiences. Moreover, SmarterContext monitors changes in favorite product and service categories at runtime. Therefore, rather than suggesting deals based only on the categories manually registered by the user, our recommendation algorithm takes advantage of up-to-date user preferences. These preferences are gathered in the form of context information from the interactions that the user has performed with different web sites. SmarterContext also provides SmarterDeals with fresh information about user locations at runtime to improve the effectiveness of recommended coupons.

To validate our approach with real data, we used the Yelp's academic data set [32] to simulate the context information gathered by SmarterContext. For this, we transformed the 271,418 product and service category ratings and 65,411 real users of Yelp's data set into Resource Description Data (RDF) [49] compliant with context models in SmarterContext. In this thesis, we present our results using a comparative analysis that involves other well known recommendation approaches.

1.3 Thesis Outline

This chapter briefly introduced our research area and our goals and motivations. The remaining sections of this thesis are organized as follows.

Chapter 2 describes in detail the problem which is to be tackled along with its context, its impact and the overall motivation for this research. It presents a thorough review on related background and prior work.

Chapter 3 presents an overall view of the proposed system as well as information

on algorithms and components involved in SmarterDeals, the new context-aware and user-centric deal recommender system.

Chapter 4 describes the data set used in this research to simulate the required context data for evaluating the proposed system.

Chapter 5 reports and analyzes the results of evaluating this system and compares it to other well-known methods.

Chapter 6 concludes this research with lessons learned and potential future work.

Chapter 2

Problem Description and Background

This chapter presents an overview of context-aware systems, recommendation algorithms, and context-aware recommender systems. The purpose of this chapter is to describe and discuss the state-of-the-art of different concepts and methods related to this research and to explain the research problem to motivating this study.

2.1 Context-Aware Systems

Advances in ubiquitous or pervasive computing technologies in today's systems offer services to users anytime and anywhere. But to provide proper services to users, applications and systems should be aware of users and their ever-changing circumstances. This capability is known as context-awareness. Context provides information about the present status of people, places, things, and devices in the environment. The status can include preferences, location, time, social relationships, and any other information that can be used to characterize the situation of an entity. When a system is context-aware, it can use contextual information of related entities and adapt

itself to the changing situations. The system should be able to extract, interpret and use contextual information and adapt its functionality to the current environment [31].

2.1.1 Context in Recommender Systems

Context is a multifaceted concept that has been studied in multiple disciplines including computer science (primarily in artificial intelligence and ubiquitous computing), cognitive science, linguistics, philosophy, psychology, and organizational sciences [5]. Context can be defined as any information useful to characterize entities that affect the situation of users [2] or more general conditions or circumstances which affect something. However, there are several definitions of context across various disciplines and their subfields. Bazire et al. present 150 different definitions of context from different fields [13]. In the field of recommender systems, context can be defined in association with data mining, e-commerce personalization, databases, information retrieval, ubiquitous and mobile context-aware systems, marketing, and management. The following subsection attempts to describe context as defined in different fields of research [5, 37].

In the data mining field, context is defined as those events which characterize the life stages of a customer and can determine a change in his/her preferences and status. Examples of these events are graduation, marriage, a new job, birth of a child, divorce and retirement. Context-awareness in data mining techniques can help to discover patterns related to specific contexts. In *e-commerce personalization*, context is defined as "intent" of a purchase by customers. For example, a customer may behave differently when buying a book for him/herself, or for his/her child as a gift. A customer of an online movie store may choose a different genre of movie depending on who is accompanying them such as friends or family. Building a sep-

arate profile for the context of each user's purchase can be beneficial to building better predictive models across different e-commerce applications. In *ubiquitous and mobile context-aware systems*, a users' location has an important role in making the systems context-aware or location-aware. Location Based Services (LBS) are widely developed and have context information such as location, time and the type of device help them to send more relevant information to their users. There have been many mobile applications developed over the last few years. Applications relating to tourism or businesses such as restaurants or theatres that advertise their services to local visitors using online coupons are examples of context-aware e-commerce applications. In *databases*, incorporating user preferences has added contextual capabilities to some database management systems. Depending on the context information associated with user preferences, these systems may return different responses or results to the same queries. Another field that contextual information can be helpful in is *information retrieval* systems. There is the possibility for these systems to base their retrieval decisions not only on queries and document collections, but also on information about search context. However, most of the current context-aware information retrieval and access techniques focus on short-term problems and immediate user interests and requests instead of long-term user tastes and preferences. In *marketing and management*, context is considered as the situation in which the transaction takes place. Customers can make different decisions, behave differently and prefer different products and services depending on their situations. According to Lilien et al. [24], "consumers vary in their decision-making rules because of the usage situation, the use of the goods or services (for family, for gift, for self) and purchase situations (catalog sale, in-store shelf selection, and sales person aided purchase)." This observation helps us understand why taking context of users into account when personalizing systems is very helpful. User-centric and personalized systems take advantage of incorporating

contextual information in these systems.

2.1.2 Obtaining Contextual Information

Context information can be obtained in different ways. One way is to obtain it explicitly(i.e., by directly asking users for information about their preferences). For example, a website can ask its users to fill a web form while signing up. This web form can gather all required contextual information for the system. Another way of explicit context gathering is adding ratings or like and dislike features. On the other hand, context information can be extracted *implicitly*. Location, time, device type, and network properties are examples of contextual information that can be obtained implicitly without interacting with the user. The other way is inferring the information using data mining techniques. For example, by mining the history of purchased items or movies watched by a family, and using well trained classifier techniques, one can infer the number, age and gender of family members.

2.1.3 Context Management Systems

As stated earlier, context-aware systems are adaptable and capable of acting on behalf of users autonomously. A large part of research and development of context-aware systems is related to context modelling, representation and reasoning that is associated with gathering, and evaluating and maintaining context information in a reusable and extendable way. One of the modelling techniques for context information is *ontology based models*. Context can be considered as a specific type of knowledge and any known framework for knowledge representation and reasoning may be appropriate for handling context. By providing formal semantics to context data, ontology-based models offer the possibility to share and integrate context among different sources [15]. Following is a brief description of SmarterContext, a dynamic context mod-

elling, representation and reasoning infrastructure that has been adapted and applied in this research.

SmarterContext [52] is a dynamic context management framework that empowers users to control the management of their personal context information. The main components of SmarterContext are (1) the SmarterContext ontology, (2) the service-oriented software infrastructure, and (3) the users personal context spheres. The SmarterContext ontology, which includes several vocabularies, supports context representation and reasoning [50]. The service-oriented infrastructure provides the software components required to manage the context information life cycle: context gathering, processing, provision, and disposal. Users personal context spheres are repositories that store the personal context data of SmarterContext users in the form of Resource Description Framework (RDF) statements [51].

Context representation and reasoning in SmarterContext is supported by the SmarterContext ontology. The SmarterContext ontology exploits RDF [16] and OWL-Lite [48] to represent context types and the relationships among them explicitly, and to infer implicit context facts from these context relationships. SmarterContext is designed as a modular ontology that supports vertical and horizontal extensibility. Its foundational module, *general context (GC)*, enables context representation and reasoning for any problem domain. Vertical extensibility makes the SmarterContext ontology applicable to different problem domains. It is realized by defining more specialized modules that inherit from the GC module or other modules derived from GC. The application of the SmarterContext ontology to a particular domain may imply the definition of several hierarchical levels. For example in the case study presented in this research, we derived the shopping module from the *personal web context (PWC)* module. Horizontal extensibility is realized by importing existing ontologies or vocabularies into any of the modules defined in the SmarterContext ontology. The

namespaces of the main modules of the SmarterContext ontology are *gc*, *pwc*, and *shopping*. Table 2.1 presents the context entity types and context relationships (i.e., object properties) that are relevant to the SmarterDeals case study.

Context Type (Class)	Description	Supertype
gc:ContextEntity	Entity. The superclass of any context type.	owl:Thing
gc:LocationContext	Entity. The place of settlement or activity of an object.	gc:ContextEntity
gc:GeoLocation	Entity. The latitude and longitude that describe a physical location.	gc:PhysicalLocation
pwc:PWESite	Entity. Any web site compliant with SmarterContext - e.g., an on-line store.	pwc:WebResource
pwc:User	Entity. Any person registered into SmarterContext.	gc:HumanEntity
shopping:Product ServiceCategory	Entity. A product or service category offered or advertised on-line - e.g., American restaurants.	pwc:WebEntity
gc:locatedIn	Object property. Its value represents the location where the subject (an IndividualContext or LocationContext entity) is located.	gc:location Relationship
pwc:hasIntegrated	Object property. Its value represents a context entity that has been integrated into a personal context sphere.	gc:association Relationship
pwc:preferredLocation	Object property. Its value defines the preferred location of a user.	gc:location Relationship
pwc:ranked	Object property. An interaction to denote that the user has given a ranking value to a context entity represented by the object.	pwc:userInteraction
shopping:related ProductOrService	Object property. Denotes that two product or service categories are related to each other.	gc:association Relationship

Table 2.1: Context entities and Context relationships of the SmarterContext ontology that are relevant to the SmarterDeals application.

2.2 Recommender Systems

Recommendation Systems (RSs) are software applications and tools that provide recommendations to users for various items. "Item" is a general term used in recommender systems and denotes what the system recommends to users [23]. Depending on the purpose of a recommendation system, the item may be books, music, movies, news, or any other product or service. There has been significant research and work done in both academia and industry in this area since the mid-1990s and over the last decade, but it is still an interesting field of research because of the overload of information in the digital world [4]. By providing personalized recommendations, recommender systems help users find the products and services that they might be more interested in with less effort. Although, due to the fast growth of available digital information and technology in today's world, there is still tremendous need and space for improvements and optimizations in recommendation techniques, and also there can be many more application areas covered by these systems. The subsequent sections summarize the state-of-the-art, classifications and possible extensions of the recommender systems.

2.2.1 Recommendation Techniques

In general, recommendation techniques are reduced to estimating ratings for items that are not used or seen or rated by a user. The process of predicting these ratings is usually based on the previous ratings of the user. Once we have estimated the unknown ratings for a user, we can recommend the items with the highest predicted ratings [4]. To perform this task, recommendation systems need an input usually illustrated as a user-item rating matrix which shows users' preferences. Table 2.2 illustrates subsets of such an input to a recommendation system. The task of the

	item1	item2	item3	item4
user1	3	5	4	4
user2	-	2	3	5
user3	4	-	3	5
user4	3	4	3	-

Table 2.2: User-Item Rating Matrix

recommender system is to estimate or predict the unknown ratings shown by "-" in the table. Once the system predicted the ratings for items not rated by a user, it can recommend the *Top-N* items that received the highest estimated ratings.

A more formal formulation of the recommendation problem is as follows [4]: Let C be the set of all users and S be the set of all items. Both user and item spaces can be very large, considering millions of users and available items in applications such as movie, CD, or news recommender systems. Let u be a *utility function* measuring usefulness of item s to user c, i.e., u: $C \times S \rightarrow R$, where R is a set of real or non-negative integers within a certain range which is usually represented by ratings. Then, for each user $c \in C$, the goal of the recommender system is to find $s' \in S$ that maximizes the user's utility.

There are many different ways to estimate unknown ratings based on various techniques from machine learning, approximation theory and various heuristics. Generally, recommender systems are classified according to their approach to rating estimation [4] and, usually, they are classified as below and explained further in the following subsection:

1. *Content-based recommendations:* Items recommended to the user are chosen according to similarity to user's preferences and liked items in the past;

2. *Collaborative recommendations:* Items recommended to the user are selected according to items preferred in the past by like-minded and similar users to the

user;

3. *Hybrid approaches:* Combination of content-based and collaborative filtering methods.

Content-based Methods

In content-based approaches, to recommend item s to user c, the similarity between the content of s and other items previously rated by c is measured. For example if news items related to science and technology are highly rated by a user in a news recommender system, then other news items in the field of science and technology get a higher similarity score according to this user's preferences and will be recommended to the user. To calculate the mentioned similarities, all items need to be represented by a *profile* that describes the properties of these items. These properties are usually extracted from the item's content which should be in an understandable format for machines. Text-based items such as articles, books and news are good examples of such items and this is why most of the content-based recommender systems are applications with a focus on text-related products or services.

A formal formulation of content-based recommendation techniques is explained as follows [4]: Let *Profile(c)* be the profile of user c representing the preferences of this user. This profile is obtained by analyzing the content of the items previously rated by c and is usually described by *keywords* using keyword analysis techniques from information retrieval. Let *Content(s)* be the profile of item s representing a set of attributes characterizing this item and described by *keywords*. Then the utility function *u(c, s)* is defined as *u(c, s) = score(Profile(c),Content(s))*.

As mentioned above, to describe both *Profile(c)* of user c and *Content(s)* of item s we can use keyword representations. We need to find a way to measure the "importance" of a word considering a user profile or an item content to represent the

user profile or item content by the most important words or keywords. Assume that each user profile or item content can be represented by a document d_j. The importance (or informativeness) of word k_i in document d_j is determined with some weighting measure w_{ij} that can be defined in several different ways, but one of the most popular measures for specifying keyword weights in information retrieval is the *term frequency/inverse document frequency (TF-IDF)* measure [4, 43], which is:

Assume that N is the total number of documents that can be recommended to users, keyword k_i appears in n_i of them and $f_{i,j}$ is the number of times keyword k_i appears in document d_j.

Term Frequency (TF): The term frequency or normalized frequency of keyword k_i in document d_j, is defined as:

$$TF_{i,j} = \frac{f_{i,j}}{max_z f_{z,j}} \tag{2.1}$$

where the maximum is obtained over the frequencies $f_{z,j}$ of all keywords k_z that appear in d_j.

Inverse Document Frequency (IDF): Let N be the total number of documents and n_i be the number of documents containing keyword k_i. IDF of keyword i is defined as:

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

Because the keywords that appear in many documents are not as informative and useful in distinguishing between a relevant and a non-relevant document, we penalize these with *IDF*.

TF-IDF: The TF-IDF weight or importance for keyword k_i in document d_j is defined as:

$$w_{i,j} = TF_{i,j} \times IDF_i \tag{2.3}$$

Now, we can define the content of document d_j using its keywords or important words as *Content(d_j) = (w_{1j}, ..., w_{kj})*. As we stated earlier, this document d_j can be a user profile (Profile(c)) or item content (Content(s)) represented as TF-IDF vectors $\vec{w}_c$ and $\vec{w}_s$ of keyword weights. To recommend items similar to the preferred items by the user in the past, we can easily measure the similarity between documents represented by keywords as mentioned above. So, if we consider the user profile as a document of representative keywords, and also if we consider each item as a document of important keywords, we calculate the similarity between these documents and the documents with the highest similarities are most similar to the user profile or preferences and will be recommended to the user. Using cosine similarity [7], we define the utility function as:

$$u(c,s) = cos(\vec{w}_c, \vec{w}_s) = \frac{\vec{w}_c.\vec{w}_s}{\|\vec{w}_c\|_2 \times \|\vec{w}_s\|_2} \tag{2.4}$$

There are some limitations associated with content-based recommendation methods. One of the limitations of content-based techniques for recommendation is that they need to deal with items with machine readable and understandable contents such as text-based items to be able to parse and extract item properties and features. Otherwise, they need some resources to manually manage this task but it is not practical to assign features and attributes to items by hand due to limitation of resources and existence of millions of items. For example, this limitation can make content-based recommendation approaches for video or audio based items impractical or impossible. Also, these techniques can not distinguish between good and bad articles, for example between two texts, both related to the same subject and hence using

very similar keywords, content-based recommendation methods are not able to give priority to the well-written one compared to the poor one [46]. Another limitation associated with these methods is over-specialization. It means that users are limited to recommendations related or similar to the items they liked in the past and they don't get a chance to receive recommendations of items in other tastes that might be interesting for them. Also, as a constraint of content-based techniques we can mention the new-user problem, because to make a system able to understand the user's taste and preferences to recommend relative items to him/her, a user has to rate a sufficient number of items in advance. In other words, the system can not understand preferences of a new user without enough ratings to recommend the similar items to the user's profile to him/her [4].

Collaborative Methods

Unlike content-based recommendation methods that recommend items similar to the items liked or highly rated in the past by the user, Collaborative Filtering (CF) methods recommend items liked or preferred by other users who have similar preferences to the user. It is the most popular approach of recommendation techniques, developed in academia and industry, because it addresses many of the limitations associated with content-based methods. To define it more formally, the estimation of utility $u(c, s)$ of item s for user c is based on the utilities $u(c_j, s)$ assigned to item s by those users $c_j \epsilon C$ who are similar to user c [4]. As the earliest work in this field we can mention the Grundy [42] and Tapestry systems [25] and as the first works in "automation" of prediction in collaborative filtering techniques we can mention GroupLens [41, 34], Video Recommender [30] and Ringo [46].

According to [17] collaborative filtering approaches can be classified into two general categories: memory-based (or heuristic-based) and model-based techniques. The

following subsections describe each of these groups in detail.

Memory-based CF Methods:

User-based and *Item-based* collaborative filtering approaches are typical examples of this class. These methods use the user-item rating data to measure similarity between users or items. In the user-based approach the value of an unknown rating $r_{c,s}$ by user c for item s can be calculated as an aggregate of the ratings by other users (most similar users to user c) for item s [4]:

$$r_{c,s} = \underset{c' \epsilon \hat{C}}{aggr}\, r_{c',s} \tag{2.5}$$

where $\hat{C}$ is the set of N users that are most similar to user c and who have rated item s. The simplest aggregation function that can be used is a simple average:

$$r_{c,s} = \frac{1}{N} \sum_{c' \epsilon \hat{C}} r_{c',s} \tag{2.6}$$

The most common aggregation approach is to use the weighted sum:

$$r_{c,s} = k \sum_{c' \epsilon \hat{C}} sim(c, c') \times r_{c',s} \tag{2.7}$$

where *sim(c,c')* is the similarity between users c and c'. The limitation of weighted sum is that it does not take into account the fact that different users may use different rating scales. To overcome this limitation, we can consider deviations from the average rating of users, $\bar{r}$, by using *adjusted* weighted sum [41, 44]:

$$r_{c,s} = \bar{r}_c + k \sum_{c' \epsilon \hat{C}} sim(c, c') \times (r_{c',s} - \bar{r}_{c'}) \tag{2.8}$$

where k is a normalizing factor and is usually calculated as $k = 1/\sum_{c' \epsilon \hat{C}} |sim(c, c')|$.

There are various approaches to measure the similarity *sim(c,c′)* between users or between items in collaborative filtering systems. The similarity between two users is based on their ratings of items co-rated by both users and the similarity between two items is based on the ratings given by users to them. The two most popular approaches are correlation-based and cosine-based similarity calculation methods.

Correlation-based approach: Assume $S_{cc'}$ to be the set of all items co-rated by both users c and c', then in the user-based correlation approach, the Pearson correlation coefficient [41, 46] is used to measure the similarity between users c and c' as follows:

$$sim(c, c') = \frac{\sum_{s \epsilon S_{cc'}} (r_{c,s} - \bar{r}_c)(r_{c',s} - \bar{r}_{c'})}{\sqrt{\sum_{s \epsilon S_{cc'}} (r_{c,s} - \bar{r}_c)^2 \sum_{s \epsilon S_{cc'}} (r_{c',s} - \bar{r}_{c'})^2}} \tag{2.9}$$

Cosine-based approach: In the cosine-based approach [17, 44], the two users c and c' are considered as two vectors in n-dimensional space, where $n = |S_{cc'}|$:

$$sim(c, c') = cos(\vec{c}, \vec{c'}) = \frac{\vec{c}.\vec{c'}}{\|\vec{w_c}\|_2 \times \|\vec{w_{c'}}\|_2} \tag{2.10}$$

where $\vec{c}.\vec{c'}$ shows the dot-product between two vectors.

Although the user-based collaborative filtering approach is very popular, there are some fundamental challenges associated with it. Existence of millions of users in today's systems makes the real-time process of similarity calculations between all candidate neighbours impossible and creates scalability challenges. Reducing the number of candidate neighbours by any method is helpful in increasing the computational performance of these systems but, on the other hand, the quality of recommendations which is the goal of any recommendation system is limited by the number of can-

didate neighbours. By clustering users and calculating similarity between each user with only users in the same cluster we can speed up the similarity measurement task, but there is always a trade-off between speeding up these systems by reducing data and the resulting quality of recommendations.

Sarwar et al. and Deshpande et al. [44, 21] discuss that item-based algorithms can provide better computational performance than traditional user-based collaborative methods, while at the same time providing comparable or better quality than the best user-based algorithms. As mentioned earlier, item-based algorithms are similar to the user-based algorithms except that they measure the similarity between items instead. Then, items most similar to the items liked by the user in the past will be recommended to the user.

Model-based CF Methods:

Memory-based approaches consider the entire user-item rating matrix to predict ratings of unknown items, in contrast, model-based approaches use the collection of ratings to *learn* a model using statistical and machine learning techniques, which is then used to predict the unknown ratings. [14, 17] compare their respective model-based approaches with standard memory-based approaches and claim that model-based methods outperform memory-based approaches in terms of accuracy of recommendations in some applications. However, the comparison in both cases has no underlying theoretical evidence to support it.

Collaborative filtering methods do not have some of the described limitations associated with content-based methods. They can recommend any type of items based on the items highly rated by most similar users and are not limited to text-based items. Also, their recommendations are not limited to a certain category of items liked in the past by the user because like-minded users to the user may have

liked items in any possible category. However, they still have the new user problem as in content-based methods. Because they need a sufficient number of ratings by the user to understand user preferences and look for most similar users to the user. There is also the new item problem in these systems when a new item is added to the system which happens regularly. The system needs a sufficient number of ratings for an item by users to be able to recommend it. Both of these named problems can be resolved using *hybrid* methods which are described in next subsection.

One of the famous problems associated with collaborative filtering methods is *sparsity*. In today's systems, there are millions of users and items which can be illustrated with a huge matrix, but the number of available ratings in the matrix compared to the number of unknown ratings is very small. This fact makes the accurate and effective ratings prediction very difficult. Items with only a few ratings do not get a chance to be recommended even if those few ratings for these items are very high. Also, users with unusual tastes can not be matched with other users according to preferences [8].

An extension of traditional collaborative filtering techniques to overcome the sparsity problem is demographic filtering [39]. In this method the similarity measurement between users is based not only on their common ratings, but also their profiles. For example, features such as location, occupation, education, gender, and age can be extracted from user profiles and, in addition to similar ratings for co-rated items, belonging to the same demographic segment is also considered for similarity measurements.

Hybrid Methods

Hybrid methods combine collaborative and content-based methods to overcome certain limitations of each of these systems [8, 12, 20, 39, 45]. There are different ways to

combine collaborative and content-based approaches into a hybrid recommendation system categorized as: (1) combining the prediction results of each of these methods that have been implemented separately, (2) incorporating collaborative characteristics into a content-based approach or vice versa, (3) using a general unifying model that incorporates both content-based and collaborative characteristics [4].

2.2.2 Advances in Collaborative Filtering

In October 2006, Netflix, an online DVD rental service, launched an open competition (Netflix Prize) for the best collaborative filtering algorithm to predict ratings for films based on previous ratings. This competition attracted thousands of scientists, engineers and students to this field and made a lot of progress in improving collaborative filtering techniques. A real industrial data set of 100 million movie ratings was available to researchers for the first time and encouraged them to develop many different techniques rapidly, looking for an optimized collaborative filtering algorithm [36]. In September 2009, the grand prize was awarded to the "BellKor's Pragmatic Chaos" team who achieved 10% improvement over the original algorithm by Netflix. Yehuda Koren and Robert Bell, two researchers from the winning team have explained advanced techniques in collaborative filtering in a book chapter [36], a survey about recent progress in the field. They describe the first choice for implementing collaborative filtering recommender systems, *Matrix factorization* technique and also *neighborhood* methods, another popular technique in this field along with several extensions and recent innovations. The following subsections explain their advanced techniques associated with neighborhood methods of collaborative filtering algorithms which have been developed and optimized in this research.

Baseline Predictors

As stated earlier, collaborative filtering methods try to understand users' preferences with the help of rating values they give to different items. In other words, they try to capture the interaction between users and items which result in ratings. However, a part of each interaction between users and items are affected by effects associated with either users or items. For example, in a movie recommender system, there are some tendencies for some users to give higher or lower ratings because the scale of rating values in different users is different. Also, bestseller movies that are better than an average movie tend to receive higher than average rating values. These are effects associated with users or items that need to be removed from the rating values to generate ratings that represent the interaction between users and items in a more pure and unaffected way. Formula 2.11 is a measure to encapsulate those effects, which do not involve user-item interaction, within the baseline predictors (i.e., also known as biases) [36].

$$b_{ui} = \mu + b_u + b_i \tag{2.11}$$

where μ denotes the overall average rating, b_{ui} is the baseline prediction for an unknown rating r_{ui} which is the unknown rating value of item i by user u. The parameters b_u and b_i indicate the observed deviations of user u and item i, respectively, from the average. For example, suppose that we want a baseline predictor for the unknown rating of the book "Harry Potter" by user Suzan. Suppose that the average rating over all movies, μ, in this system is 3.5 stars. Harry Potter is better than an average book, so it tends to be rated 0.5 stars above the average. On the other hand, Suzan is generous in ratings, who tends to rate 0.5 stars higher than the average. Thus, the baseline predictor for Harry Potters rating by Suzan would be 3.5 + 0.5 +

$0.5 = 4.5$ stars.

There are different ways to estimate b_u and b_i. A simpler but less accurate way of such estimation is described below:

First, for each item i calculate:

$$b_i = \frac{\sum_{u \epsilon R(i)} (r_{ui} - \mu)}{\lambda_2 + |R(i)|} \tag{2.12}$$

Then, for each user u calculate:

$$b_u = \frac{\sum_{i \epsilon R(u)} (r_{ui} - \mu - b_i)}{\lambda_3 + |R(u)|} \tag{2.13}$$

where *R(u)* denotes all the items for which ratings by user u are available and *R(i)* denotes the set of users who rated item i. The regularization parameters, $\lambda 2$ and $\lambda 3$, which are determined by cross validation for the Netflix dataset are: $\lambda 2 = 25$ and $\lambda 3 = 10$.

Neighborhood Models

The most common approach to collaborative filtering is based on neighborhood models. Its original form is user-based which is based on user-user similarity measurements. Later item-based methods were introduced which are based on item-item similarity calculations. [36] calculates item-item similarities for an item-based approach using the Pearson Correlation Coefficient (Formula 2.9). Once they calculated the similarity between items, they needed a similarity-based neighborhood method which predicts the unknown rating r_{ui} of item i by user u based on the similarity between i and previously rated items by u. They present the following method as the most popular approach to neighborhood modeling and collaborative filtering. The goal is to predict r_{ui} which is the unknown rating by user u for item i. Using the similarity

measure, they identify the k most similar items to i rated by u. This set of k neighbors is denoted by $S^k(i;u)$. Then, they predict the value of r_{ui} as a weighted average of the ratings of neighboring items, while adjusting for user and item effects through the baseline predictors:

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{j \epsilon S^k_{(i;u)}} S_{ij}(r_{uj} - b_{uj})}{\sum_{j \epsilon S^k_{(i;u)}} S_{ij}} \tag{2.14}$$

where $\hat{r}_{ui}$ is the predicted value for r_{ui}. Section 3.5 describes how we adapted and optimized a user-based approach of their method as a context-aware collaborative filtering technique.

2.3 Context-Aware Recommender Systems (CARS)

As described in the previous sections, there has been much work done in the field of recommender systems. However, in many applications considering only users, items, and ratings are insufficient for suggesting relevant recommendations and it is necessary to consider and incorporate relevant contextual information in these systems. In some applications it is important *when, where, and with whom* the user is using the application. Therefore, relevant recommendations should be based on this context information of the user. Decision making in selecting a movie by a user can be dependant on the day of the week (e.g., weekday, weekend), the place they will watch it (e.g., theatre, home) and accompanying people (e.g., family, friends, children). So, unless a movie recommender system knows about the context of a user, it may provide poor and non relevant recommendations to the user. Another example is a "smart" shopping cart that *holds bread, butter and preferences* [53] providing real-time recommendations to shoppers using wireless location-based technologies. This application needs to take into account not only information about products and customers but

also such contextual information as shopping date/time, store, who accompanies the primary shopper (e.g., children), products already placed into the shopping cart, and its location within the store. Adomavicius et al. discuss that accurate prediction of customer preferences assuredly depends upon the degree to which we have incorporated the relevant contextual information into a recommendation system [3]. The following subsection provides a brief overview of prior work in this research area.

2.3.1 Prior Work on Context-Aware Recommender Systems

The *multidimensional data* (MD) model proposed by Adomavicius et al. [3] is the most referenced work in the area of context-aware recommendation systems. In the MD model contextual information is defined using classical OLAP (On-Line Analytical Processing) [18] hierarchies which are widely used in data warehousing applications in the database community. To provide recommendations based on contextual information, it presents a *multidimensional* recommendation model that makes recommendations based on multiple dimensions and, therefore, extends the classical two-dimensional user-item approach. In the classical approach, once the recommendation process is provided with an initial set of ratings that is either explicitly obtained from users or is implicitly inferred by the system, the recommender system tries to estimate the rating function R for the unknown ratings of (user, item) pairs, $R : User \times Item \rightarrow Rating$. This is called a two-dimensional approach since the only considered dimensions in the recommendation process are *User* and *Item*. Whereas, in the multidimensional approach, contextual information is incorporated into the recommendation process as an additional dimension and the rating function estimation is performed over a multidimensional space, $R : User \times Item \times Context \rightarrow Rating$. The authors discuss that using a reduction-based method for rating estimation by applying a traditional collaborative filtering technique only on relevant contextual segments,

we can reduce the multidimensional approach to the well-known two-dimensional approach. Also, they demonstrate that in most situations, considering contextual information provides better recommendations on some contextual segments. However, it exhibits worse performance on some other segments. Hence, they propose a combined reduction-based approach that verifies the contextual segments on which CF outperforms the context-aware approach and applies the CF method on those segments while applying the context-aware method on other segments. The authors have written a book chapter [5] containing a thorough review of context-aware recommendation approaches in which they categorize context-aware recommender systems into three categories: *contextual pre-filtering*, *contextual post-filtering* and *contextual modelling*. In the contextual pre-filtering approach, the contextual information is used as a label for filtering out those ratings that do not correspond to the specified contextual information. So, before launching the main recommendation method, the unrelated ratings in the sense of context are filtered out and the recommendation method is launched on the remaining reduced dataset. [9, 10] are examples of contextual pre-filtering. Whereas, in the contextual post-filtering approach, after launching the traditional recommendation method on the dataset and producing a list of recommendations, contextual information is used to contextualize the obtained recommendations. In contextual modelling, context is integrated directly into the model. The approach described in [33] belongs to the latter category.

Baltrunas et al. [11] propose an item splitting method which extends the traditional CF data model by assuming that each rating r in a users-items matrix is stored together with a nominal piece of context information c. The proposed method identifies items in the matrix which have significant differences in the ratings and then for each one of these items, they split its ratings into two subsets, creating two new artificial items with ratings belonging to these two subsets. They explain that splitting

is beneficial if the ratings in the newly created items are more homogenous, or if the ratings in the two newly created items are statistically different. Later, when they want to make a prediction for user u and item i which has been split into artificial items $i1$ and $i2$, where $i1(i2)$ contains ratings for item i acquired in the contextual condition $c1(c2)$, the prediction is computed for the item $i1(i2)$ if the current context of user u is equal to $c1(c2)$.

Panniello et al. [38] apply contextual pre-filtering and post-filtering methods as defined in [3] on two datasets and compare the results. They use two different post-filtering methods, Weight and Filter, as well as the exact pre-filtering method and apply them on the two datasets. Their results show that the Filter post-filtering method outperforms exact pre-filtering and that exact pre-filtering outperforms the Weight post-filtering method on two datasets. In particular, they propose that to choose between applying pre-filtering and post-filtering methods on a specific application, first, to compare pre-filtering with the un-contextual filtering method, if the un-contextual case outperforms the pre-filtering, then they argue that a good choice in the post-filtering approach is better than pre-filtering.

Karatzoglou et al. [33] present a model, called Multiverse Recommendation, in which a model-based Collaborative Filtering method based on Tensor Factorization, a generalization of Matrix Factorization, integrates contextual information with the system in a flexible and generic way. This contextual integration is done by modelling the data as a User-Item-Context N-dimensional tensor instead of the classic 2D User-Item matrix. They compare their method to non-contextual Matrix Factorization and also to two state-of-the-art context-aware methods (OLAP [3] and item-splitting [11]) and show that Tensor Factorization outperforms them in terms of the Mean Absolute Error (MAE). However, Rendle et al. [40] argue that although Multiverse Recommendation [33] based on the Tucker tensor factorization model is the best performing

method in terms of predictive accuracy for context-aware rating prediction, its model complexity is exponential in the number of context variables and polynomial in the size of the factorization. They propose a fast context-aware rating prediction method by applying Factorization Machines (FM) to model contextual information and show that their approach outperforms Multiverse Recommendation in prediction quality and runtime.

Based on memory models developed in cognitive science, Anand et al. [6] propose a user model consisting of short term memory (STM) which stores current interactions of the active user and long term memory (LTM) which stores previous interactions containing user's ratings as well as the context in which ratings have been acquired. They hypothesize that enrichment of STM with contextually relevant ratings extracted from LTM can produce more accurate and contextual recommendations. Similarly, Abbar et al. [1] propose that in addition to the user profile, they consider context which is a set of features characterizing the environment within which users interact with the recommendation system in their recommendation approach. Their contextual data includes date, time, location and browser, system and device properties. Their key idea is to base the top k neighbours detection only on the profile parts relevant to a given context instead of a whole user profile. They describe that before comparing users to find the top k neighbours, profiles of all users are filtered and adapted to the current context of the active user. So, candidate neighbours are users who rated the item to be recommended in a context similar to the active context. They explain that although there are some similarities between their approach and the one in [6], considering a representational view of context instead of its interactional view separates these two approaches.

Baltrunas et al. provide a general architecture of context-aware recommender systems and analyze separate components of this model [9]. In their proposed ar-

chitecture, there is a component named model adapter which is responsible for integrating contextual data into the prediction algorithm. They describe item selection as an example of implementation of the model adapter. In item selection, they consider that some items are more important than others for making predictions. The context of the item to be recommended plays the role of the context descriptor and the predictive items are dynamically selected with respect to it. They describe their algorithm for item selection as follows: to generate a prediction for user u and item i, the first step is to find k nearest neighbours of u. Instead of finding these neighbours by considering all of the ratings, we find only f items which are most similar to i and are co-rated by users, then we compute user-to-user similarities using only the ratings related to these f items which are most similar to i.

Baltrunas et al. introduce a context-aware (time-aware) recommendation approach called user micro-profiling, in which they split each single user profile into several possibly overlapping sub-profiles, each representing users in particular contexts (times) [10]. The predictions are done using these micro-profiles instead of a single user model. In another work, Chen et al. propose to capture and store a snapshot of the context (e.g., location, weather) along with each user-rating [19]. It can be acquired from either the embedded sensors in the mobile device itself, or a smart environment. Then, they incorporate context into predictions by making ratings weighted with respect to the similarity between context x in which the rating r was given and the context c of the active user.

The next chapter explains our context-aware recommendation approach using dynamically managed contextual information.

Chapter 3

SmarterDeals

3.1 Overview

Figure 3.1 presents an overview of SmarterDeals, our approach to user-centric context-aware daily-deal recommendations. Our application is composed of two main artifacts, the *recommendation engine*, and the *filtering and personalization module*. For SmarterContext to provide SmarterDeals with personal context information, users must integrate SmarterDeals into their personal context spheres. After completing this prerequisite, SmarterContext framework provides SmarterDeals with personal context information about the user's product and service preferences, and locations.

Our recommendation engine exploits context information about the user's product or service preferences to predict daily-deal categories relevant to the user as follows. In the first step our recommendation algorithm correlates similarities among users based on the *Pearson Correlation Coefficient* (PCC) ranging from -1, which indicates a negative correlation, to +1, which indicates a positive correlation between two users and a value of 0 indicates no correlation. We consider a certain value (0.7) as the

threshold that defines a positive correlation between two users. That is, users who have a PCC equal to or greater than 0.7 are similar enough in our approach. In the second step, our algorithm aggregates the ratings of product or service categories given by similar users to the user who will receive the recommendation, to predict the rating of the corresponding product or service category according to this user. SmarterDeals decides whether a product or service category is relevant to a user using the predicted rating of the corresponding category. In our case study ratings range from 1 to 5, where 1 indicates the lowest level of interest and 5 the highest. Our algorithm recommends categories with predicted ratings equal to or greater than 4. The recommendation engine calculates the set of potential relevant categories using rating prediction for a user. In the third step SmarterDeals uses Groupon's application programming interface (API) to provide the user with business daily-deal offers according to the user's location. Then, the filtering and personalization module filters these deals according to the user's preferred categories as well as computed recommending categories for this user. Finally, SmarterDeals presents the set of potential relevant deals to the user.

SmarterDeals, supported by SmarterContext, improves the relevance of daily product and service recommendations delivered to users by exploiting:

- up-to-date product and service categories gathered from web interactions performed by the users throughout their web experiences, and
- up-to-date information about current and preferred users' locations.

As presented in Figure 3.1, SmarterContext and Groupon API are external components of our system. In following subsections, we introduce these external components and then we describe the activity flow of the internal components, the *recommendation engine* and the *filtering and personalization module*.

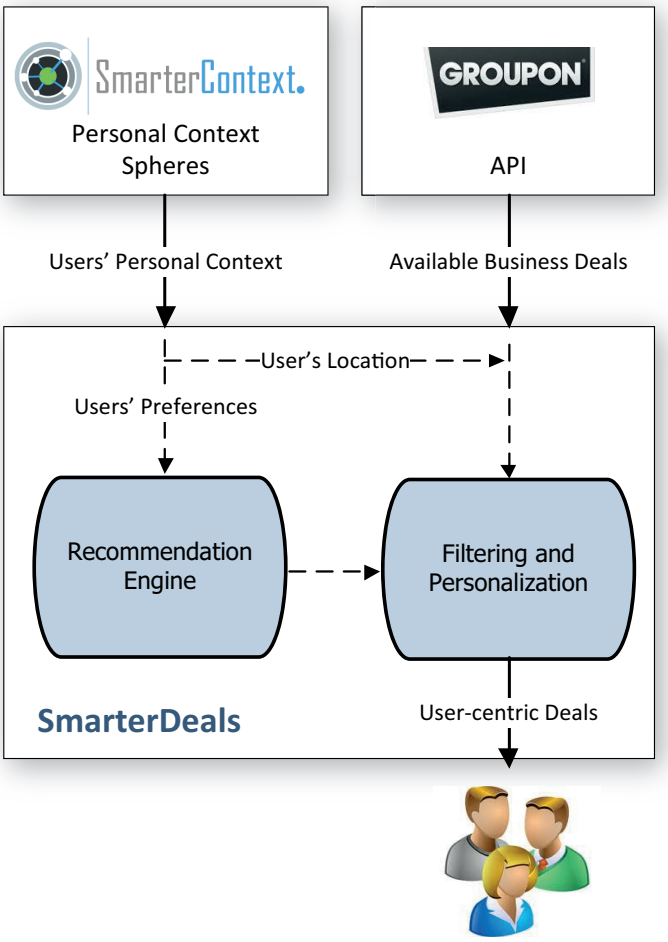

Figure 3.1: SmarterDeals overview

3.2 Context Management with SmarterContext

A prerequisite for SmarterContext to manage a user's context information is the creation of the user's personal context sphere. For this, users register themselves into the SmarterContext framework by providing some personal context information such as age, gender, preferred location, and preferred payment methods. They may also decide to register web sites or applications compliant with SmarterContext. That is, applications instrumented to interchange context information with the SmarterContext infrastructure. The gathering of context information from context sources, and its provisioning to web applications is controlled

by the user. Consequently, SmarterContext interchanges personal context only with those applications registered by the user into their personal context sphere.

SmarterContext deploys sensors at the user side to identify context sources and providers. For example, through its browser extension, SmarterContext detects whether web applications accessed by the user are enabled to communicate with SmarterContext. If so, SmarterContext checks whether this application is already registered into the user's context sphere. If it is registered, SmarterContext provides this application with relevant context information about the user. Otherwise, our context manager asks the user whether she wants to integrate this new application into her personal context sphere.

One of the most relevant mechanisms of SmarterContext for gathering context information is based on the monitoring of users' web interactions. In a smarter commerce case study presented in [52], simple RDF sensors deployed at the context provider side, e.g., an on-line shopping application, keep track of "likes", "wishes", "rankings", and "purchases" interactions performed by the user. From these interactions SmarterContext understands what product and service categories are interesting to the user. Figure 3.2 presents the RDF graph representation of a user's ranking interaction gathered by SmarterContext. RDF/XML serializations constitute the interoperability mechanism between the SmarterContext infrastructure and context providers and consumers.

As stated earlier, for SmarterContext to provide SmarterDeals with personal context information, users must integrate SmarterDeals into their personal context spheres. After completing this prerequisite, SmarterContext framework provides SmarterDeals with personal context information about the user's product and service preferences, and locations.

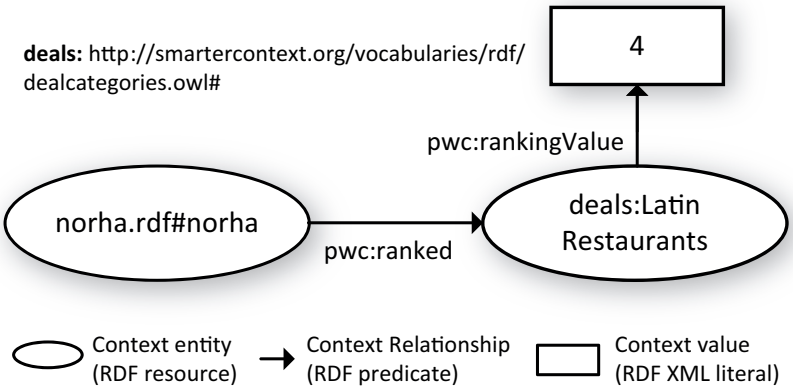

Figure 3.2: The RDF graph representation of a user's ranking interaction.

3.3 Daily Deals with Groupon

Groupon delivers daily coupons based on the personal information registered by the user during the sign up process. This information corresponds to the user's favourite locations, gender, age, and favourite deal categories. Users can edit their personal information at any time through Groupon's web or mobile applications. Groupon allows users to share deal recommendations by email, as well as to broadcast them to their social networks.

Despite Groupon's success, the current implementation of its daily-deal application can frustrate users with irrelevant deals due to stale profiles. Groupon delivers offers of products and services taking into account only the information registered by the user during the sign up process. Nevertheless, most of this information becomes out-of-date quickly. In daily-deal applications, location and preferred deal categories are types of highly dynamic context information. With respect to the user's preferred locations, Groupon delivers deals related to the whole set of registered locations, which can include different cities. This practice lacks location-aware filtering mechanisms thus compromising the effectiveness of delivered coupons. For example, for users who are frequent travellers, daily deals must be delivered taking into account the

current user's location, even if this location is not part of the user's list of favourite locations. Regarding the list of preferred deal categories, it is ineffective to send coupons using only the information registered during the sign up process, since the relevance of deal categories is highly dependent on changing context information such as time. Hence, categories that could have been relevant yesterday, may be no longer relevant today nor in the near future. For example, a user's young children may be interested in children books today, but probably not in three or four years. In Groupon, users must change their personal information and preferences manually to maintain the relevance of received offers.

Groupon has an Application Programming Interface (API) [27] for developers to integrate Groupon into their systems. The Groupon public API includes two different services. A Divisons API that provides information about all the cities where Groupon offers deals; and a Deals API that provides information about the current deals in a particular location. The divisions of Groupon are currently based around metropolitan areas, but Groupon expects to expand regionally in the future. Developers can receive data in either XML or JSON format, and need to sign up for an API key.

3.4 Activity Flow Description

To integrate Groupon in our system we take advantage of both APIs of Groupon. We use the Divisons API to get information about all the cities where Groupon offers deals. Therefore, when a user uses our system to receive personalized deals, the system maps the user's location provided by SmarterContext to the nearest city where Groupon is active. This can be either the same user's location for users living in large cities or the nearest metropolitan area to them for users living in

cities where GROUPON does not offer deals. Then, we send a request to the other GROUPON's resource, Deals API, to receive the current deals for this mapped location by embedding the location in the request. We chose to receive deals list in JSON format which is the default format of GROUPON response, rather than receiving it in XML format.

Now that we have a list of current deals for a user considering his/her location, we are ready to parse and filter it based on the categories of offered products or services to personalize it with respect to the user's preferences. In the information about each deal, there is a field named "tag" which states the product/service category of the deal. GROUPON has 633 product/service categories which are subcategories of 18 super-categories called *parent categories* hereafter. [1] SMARTERCONTEXT uses Web Ontology Language (OWL) [48], a W3C standard to build vocabularies, or ontologies for the web. We extended the SMARTERCONTEXT ontology to include GROUPON product/service categories. In the defined ontology, product/service categories are sub-classes of parent categories. Figures 3.3 and 3.4 illustrate the class hierarchies of the extended ontology. However, these figures do not show the last level of the hierarchy which is product/service categories due to limited space.

To personalize the list of all available deals with respect to the user's preferences, the Filtering and Personalization module takes into account first the product/service categories liked or highly rated by the user and second, the product/service categories recommended to the user by the recommendation engine. Like most of other recommender systems, the input of the recommendation engine is a set of ratings by users for the items. Here, the set of ratings by users for different product/service categories is provided by SMARTERCONTEXT for both the filtering and personalization module and recommendation engine. First, the recommendation engine makes

[1] https://sites.google.com/site/grouponapiv2/api-resources/deals/Groupon_Categories.xls?attredirects=0

Figure 3.3: Class hierarchies of the extended ontology for GROUPON product/service categories

recommendations of product/service categories for users according to their previous ratings of product/service categories. Then, the filtering and personalization module filters the list of current deals based on product/service categories that are preferred by or recommended to the user. The next section presents our approach to generate context-aware recommendations using product/service category ratings.

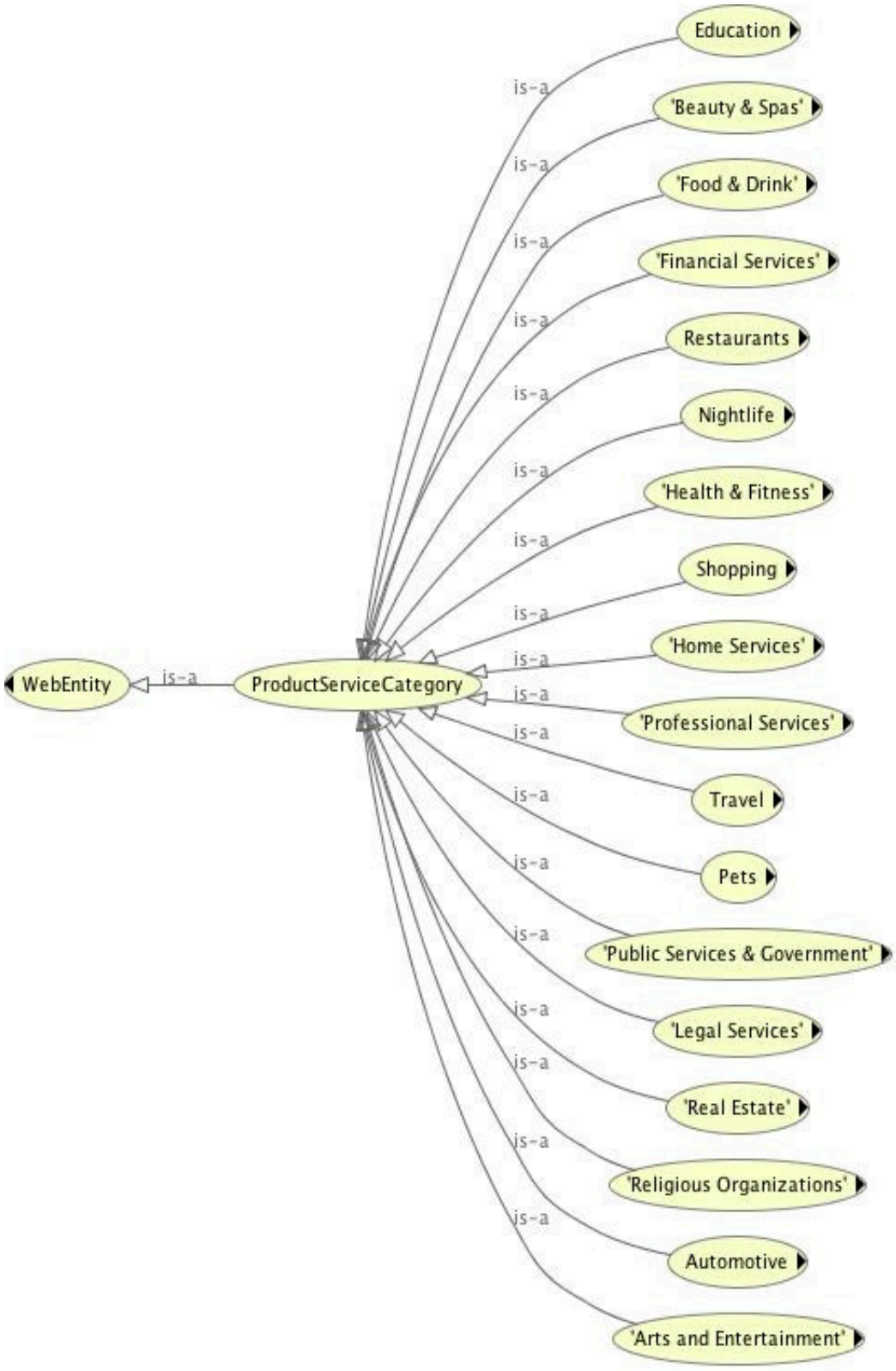

Figure 3.4: Hierarchy diagram for the extended ontology for GROUPON product/service categories

3.5 Our Approach to Context-Aware Recommendations

Our approach can be classified under *pre-filtering* context-aware recommender systems. As described in section 2.3.1, in the contextual pre-filtering approach, the contextual information is used as a label for filtering out those ratings that do not correspond to the specified contextual information. So, before launching the main recommendation method, the unrelated ratings in sense of context are filtered out and the recommendation method will be launched on the remaining reduced dataset. We believe that users behave differently in rating different product/service parent categories. For example, users having lots of fun during a nighttime activity or having an amazing sunny weekend during an outdoor activity tend to be more generous in rating these product/service categories belonging to "Nightlife" and "Arts and Entertainments" parent categories respectively. Whereas, in more sensitive situations like rating the product/service categories belonging to "Health & Fitness", "Legal Services" or "Education" they tend to be more critical and careful. Hence, we decided to consider the ratings by users belonging to each parent category separately. However, since each user may have very few or even no ratings in some parent categories, we measure user-user similarities based on the whole set of ratings by each user to avoid the sparsity problem. The following subsections explain our method in more detail.

At the first step of our user-based collaborative filtering algorithm, we calculate user-user similarities based on the whole set of data in the user-item rating matrix (see section 2.2.1). We use the Pearson Correlation Coefficient to measure the similarity between users based on the ratings of co-rated items by them.

$$sim(u,v) = \frac{\sum_{s \epsilon S_{uv}} (r_{u,s} - \bar{r}_u)(r_{v,s} - \bar{r}_v)}{\sqrt{\sum_{s \epsilon S_{uv}} (r_{u,s} - \bar{r}_u)^2 \sum_{s \epsilon S_{uv}} (r_{v,s} - \bar{r}_v)^2}} \tag{3.1}$$

where S_{uv} as the set of all items co-rated by both users u and v. Users with similarity score $>= 0.7$ are interesting for us.

Then, we apply the technique used by Bell and Koren (See section 3.5) to improve the collaborative filtering method. They explain that there are some effects associated with users or items that need to be removed from the rating values to generate ratings that represent the interaction between users and items in a more pure and unaffected way. They encapsulate those effects, which do not involve user-item interaction, within the baseline predictors called b_{ui}. We call this approach *Adjusted Collaborative Filtering (ACF)*, hereafter. Figure 3.5 illustrates their approach according to our problem.

As shown in the figure, they calculate μ which is the overall average rating. b_{uc} is the baseline prediction for an unknown rating r_{uc} which is the unknown rating value of category c by user u. The parameters b_u and b_c indicate the observed deviations of user u and category c, respectively, from the average.

$$b_{uc} = \mu + b_u + b_c \tag{3.2}$$

According to them, an option to estimate b_u and b_c is:

First, for each category c calculate category deviation:

$$b_c = \frac{\sum_{u \epsilon R(c)} (r_{uc} - \mu)}{\lambda_2 + |R(c)|} \tag{3.3}$$

Then, for each user u calculate user deviation:

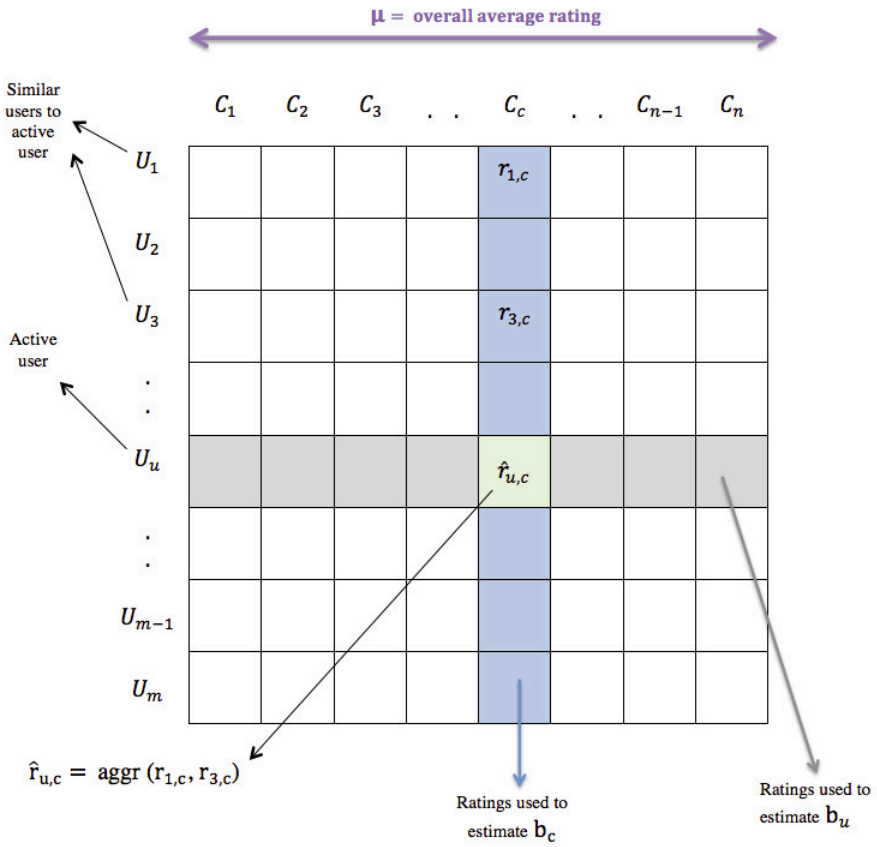

Figure 3.5: Adjusted Collaborative Filtering by Bell and Koren

$$b_u = \frac{\sum_{c \epsilon R(u)} (r_{uc} - \mu - b_c)}{\lambda_3 + |R(u)|} \tag{3.4}$$

where *R(u)* denotes all the categories for which ratings by user u are available (as shown by the coloured row in the Figure 3.5) and *R(c)* denotes the set of users who rated category c (as shown by the coloured column in the Figure 3.5). Koren and Bell demonstrated that 25 and 10 are typical values for λ_2 and λ_3, respectively. We used the same values for the Yelp data set.

Our main contribution is to incorporate contextual information in the ACF method. As we stated earlier, we propose that ratings by users should be considered separately

in different parent categories. Users show different decision making tendencies in rating categories belonging to different parent categories with respect to the nature of those categories. According to Bell and Koren, there are some effects associated with users that need to be encapsulated within the baseline predictors using observed deviations of users from the average, b_u. We propose that observed deviations of users from the average can be different for different parent categories because the quality and quantity of the effects associated with users regarding to different parent categories are not the same. Hence, for each user we estimate b_u for each parent category separately. Also, instead of a unique μ as the overall average rating, we calculate the average rating by each parent category. Figure 3.6 illustrates our approach.

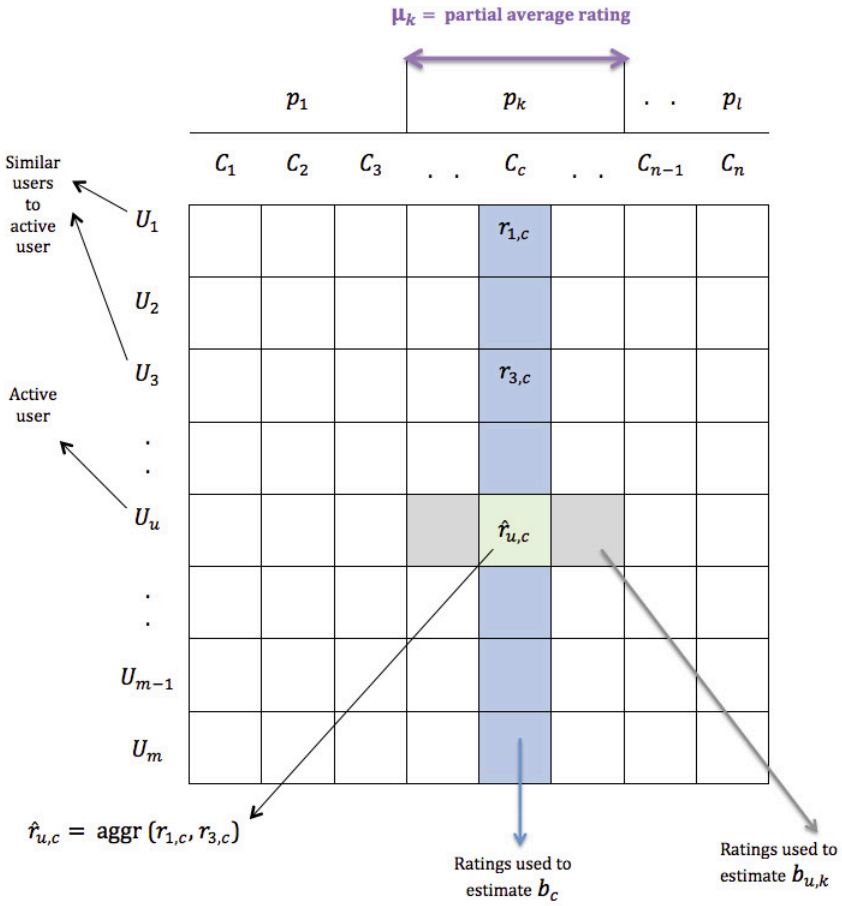

Figure 3.6: Context-aware Adjusted Collaborative Filtering

As mentioned above, we calculate the average of ratings belonging to each parent category p_k separately and denote it by μ_k. Then, we estimate category deviation b_c for each category c by calculating formula 3.3. Finally, for each user based on each parent category we estimate user deviation $b_{u,k}$ as:

$$b_{u,k} = \frac{\sum_{c \epsilon R_k(u)} (r_{uc} - \mu_k - b_c)}{\lambda_3 + |R_k(u)|} \tag{3.5}$$

where $R_k(u)$ denotes the set of categories belonging to parent category p_k rated by user u which is shown by partially coloured row in the figure 3.6.

Therefore, our baseline prediction for an unknown rating r_{uc} of category c by user u where $c \epsilon p_k$ is:

$$b_{uc} = \mu_k + b_{u,k} + b_c \tag{3.6}$$

When user similarities and baseline predictions are calculated, we can apply the neighborhood model described in 2.2.2 to predict the unknown ratings. Since we want to develop a user-based collaborative filtering method, we slightly modified formula 2.14 which is an item-based approach, as follows:

$$\hat{r}_{uc} = b_{uc} + \frac{\sum_{v \epsilon S^t_{(u)}} S_{uv}(r_{vc} - b_{vc})}{\sum_{v \epsilon S^t_{(u)}} S_{uv}} \tag{3.7}$$

where $\hat{r}_{uc}$ is the predicted value for r_{uc} and $S^t_{(u)}$ denotes the set of similar users to user u with Pearson Correlation $>= t$. Figure 3.7 represents the flow diagram of our context-aware collaborative filtering approach.

In our case study ratings range from 1 to 5, where 1 indicates the lowest level of interest and 5 the highest. Hence, we consider product/service categories with predicted rating equal to or greater than 4 as relevant categories to user and add

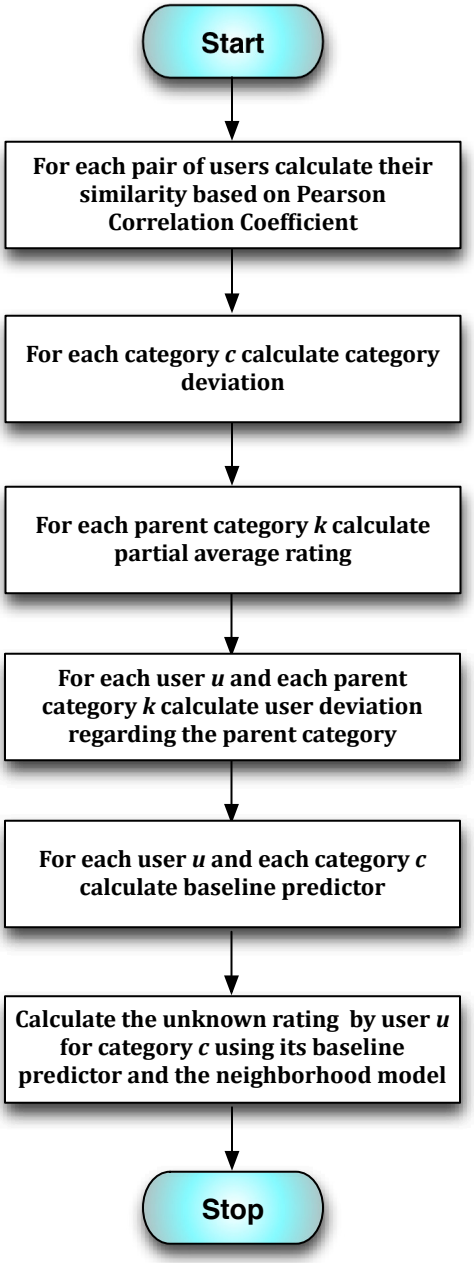

Figure 3.7: Flow Diagram of Context-aware Adjusted Collaborative Filtering

them to the user's recommendation list.

The next chapters explain our case study to evaluate our approach to context-aware recommendation filtering using a comparative analysis that involves other well known recommendation techniques.

Chapter 4

Experiments

4.1 Simulating Context Data with the Yelp Academic Dataset

Personal context spheres store personal context data in the form of Resource Description Framework (RDF) graphs. To validate our approach to smarter deal recommendations with real data, we used the Yelp Academic Dataset [32].

Yelp's dataset provides reviews of the 250 closest businesses for 30 universities for students and academics to explore and research. The data set includes 271,418 ratings that 65,411 real users have given to 6,900 local businesses. Ratings range from 1 to 5, and local businesses have been tagged with one or many of the 365 product and service categories defined by Yelp. We used the information of Yelp's data set to create 65,411 RDF graphs. Each graph simulates the context sphere of a user. Appendix A illustrates a generated graph for a user. If you pay attention to the ratings by this user which all belong to two parent categories "Health & Fitness" and "Food & Drink", you can see that the average ratings for these two parent categories are 5 and 2.66 respectively. It is a good example of how different users may behave

in rating different categories.

Since in Yelp's data set users have ranked businesses instead of products or services, we obtained favorite users' product and service categories from the categories associated to the businesses ranked by these users. Yelp's businesses are located in 67 cities across North America. Users' locations are obtained from the locations of the businesses reviewed by each user. Nevertheless, the set of cities or divisions in Groupon differs from the set of cities related to businesses in Yelp. Thus, to use Groupon's API in SmarterDeals, for each location associated to a user in Yelp, we considered all of the nearby divisions in Groupon. In this way, if Groupon is unavailable in the user's relevant locations, we can still deliver Groupon deals related to nearby locations. For this, we extended the SmarterContext vocabulary that defines geographical locations to include the locations used by Groupon and Yelp.[1]

Groupon defines 633 product and service categories classified into 18 general categories, whereas Yelp's data set contains 433 product or service categories with no hierarchies. 284 of these categories are exactly equal to those in Groupon. However, to recommend deals based on the product and service categories defined by Groupon, we mapped the remaining 149 Yelp categories into similar Groupon categories. SmarterContext classifies products and services using the Google product taxonomy [26]. For this case study, we extended this taxonomy by creating a complementary ontology, the *deals* ontology,[2] from the set of Yelp's product and service categories mapped to Groupon's deal categories along with their parent categories.

This simulation of context data provides a set of real rating data for our system including real users' locations and their ratings of product/service categories.

[1]The SmarterContext *geo* vocabulary is available at http://smartercontext.org/vocabularies/rdf/geo.rdf

[2]The *deals* ontology is available at http://smartercontext.org/vocabularies/rdf/dealcategories.owl

Therefore, we are able to evaluate our context-aware recommendation technique by applying it on this dataset to compare our result with other baseline methods. Next chapter defines our baseline methods for the purpose of comparative analysis and presents and analyzes the results.

Chapter 5

Evaluation, Analysis and Comparisons

5.1 Evaluation Method

To evaluate our approach for each existing $r_{u,c}$ rating, we create a new version of the Yelp dataset, $Y_{-r_{u,c}}$, with $r_{u,c}$ removed. Then, we apply our approach and the two baseline approaches to predict the deleted rating $r_{u,i}$. The predicted rating $\hat{r}_{u,c}$ may or may not be the same as $r_{u,c}$. In general, we are interested to see whether the error $e_{u,c} = r_{u,c} - \hat{r}_{u,c}$ is small. We repeat this procedure for each existing rating $r_{u,c}$ (i.e. we run the algorithms as many times as there are ratings in the dataset). This exhaustive evaluation gives us a precise picture of the quality of each approach. To measure the effectiveness of each approach we used *root mean squared error (RMSE)* as defined in Formula 5.1, a widely accepted metric to assess the accuracy of the values predicted by a model or an estimator with respect to the values actually observed [29] .

$$RMSE = \sqrt{\sum_{(u,c) \in TestSet} (e_{u,c})^2 / |TestSet|} \tag{5.1}$$

To conduct a comparative analysis, we selected two well-known recommendation techniques as baselines for the same test dataset and compared the calculated RMSE based on their results. The first implemented technique for comparison is traditional user-based collaborative filtering and the second one is the adjusted collaborative filtering (ACF) technique used by Bell and Koren.

5.2 Test Description and Results

To evaluate our algorithm with Yelp's dataset and since some Yelp users have a few ratings, we reduced Yelp's data set by considering only users who have at least 20 ratings. The reduced data set which is our test data set has 58,069 ratings distributed over 313 product or service categories by 1,683 users. These 313 categories, mapped to GROUPON's categories, belong to 17 parent categories instead of 18 parent categories because, by reducing the dataset the "Legal Services" Groupon parent category was deleted since it had only 11 ratings in total.

To measure user-user similarities, we applied the Pearson Collaborative Coefficient as shown by Formula 3.1 for all three recommendation methods. Then, for the ACF approach we computed baseline predictors as shown in Formulas 3.2, 3.3 and 3.4. To estimate baseline predictors as proposed by our approach, we used Formulas shown in 3.3, 3.5, 3.6. Finally, to predict the rating r_{uc} by user u for category c which has been hidden from the test dataset, for the traditional user-based collaborative filtering method we used the weighted sum aggregation function shown in Formula 2.7 and for both ACF and our approach we used the neighborhood model shown in Formula 3.7.

Because the focus of our approach is on the difference between parent categories, we designed our test in a way that at each step, it hides and predicts all the ratings

of a single parent category one at a time; computes the RMSE associated with this parent category; then continues the same process for next parent category. Hence, we are able to calculate and report RMSE based on each parent category separately and compare the performance of the methods under test. We also report the average RMSE of all parent categories for each recommendation technique. Table 5.1 presents the resulting RMSE for all three approaches. Note that parent categories shown in this figure are those having at least 50 ratings.

Table 5.1 presents our validation results. Column *Parent Category* contains the 14 GROUPON parent categories included in the reduced Yelp data set. Columns *Classic CF (C)*, *ACF (A)*, and SMARTERDEALS *(S)* present the RMSE for the two baselines—the traditional user-based collaborative filtering method and ACF approach, and our context-driven approach, respectively. Column *(AiC)*, calculated as $((C-A)/C)*100$, corresponds to the improvement of A over C. That is an example of the improvement in the error measurement (RMSE) of the ACF approach with respect to the traditional user-based recommendation method. Similarly, Column *(SiC)* represents the improvement of SMARTERDEALS (S) over the traditional method (C), and calculates as $((C-S)/C)*100$. Column *SiC-AiC* is to compare the improvement of S over C with respect to the improvement of A over C. Finally, Column *Relative Performance (RP)*, calculated as $((SiC-AiC)/SiC)*100$, represents the relative improvement of our approach with respect to ACF approach. Figure 5.1 presents the improvement in terms of accuracy of the ACF approach (AiC), and our approach (SiC), with respect to the traditional user-based collaborative filtering method (C). Figure 5.2 presents the relative performance of SMARTERDEALS with respect to the ACF approach.

Our approach is about 8.1% more accurate than classic CF and 1.3% than ACF. For about half of the categories, SMARTERDEALS is more than 2% better than ACF, with some more than 3% better.

Table 5.1: Validation Results

Parent Category	Classic CF (C)	ACF (A)	SmarterDeals (S)	(AiC)	(SiC)	SiC-AiC	Relative Performance (RP)
Automotive	1.16	1.15	1.12	1.4%	3.9%	2.5%	173.4%
Financial Services	1.67	1.60	1.55	3.9%	6.8%	2.9%	74.8%
Real Estate	1.74	1.68	1.64	3.1%	5.3%	2.3%	73.2%
Health & Fitness	1.09	1.02	0.99	6.2%	9.3%	3.0%	48.4%
Beauty & Spas	1.17	1.08	1.04	7.4%	11.0%	3.5%	47.1%
Education	1.16	1.11	1.09	4.5%	6.6%	2.1%	45.5%
Travel	0.97	0.91	0.89	6.1%	8.4%	2.3%	36.8%
Food & Drink	0.95	0.87	0.85	8.1%	10.3%	2.2%	26.7%
Arts and Entertainment	0.93	0.86	0.84	7.2%	9.0%	1.9%	26.2%
Restaurants	0.95	0.87	0.86	8.7%	9.8%	1.1%	12.7%
Shopping	1.60	1.55	1.55	3.1%	3.3%	0.2%	7.6%
Nightlife	0.85	0.76	0.75	11.6%	12.2%	0.6%	5.3%
Professional Services	0.90	0.78	0.80	12.6%	10.3%	-2.3%	-18.5%
Public Services & Government	1.11	0.98	1.03	11.8%	7.8%	-4.0%	-34.0%
Average	1.16	1.09	1.07	6.8%	8.1%	1.3%	37.5%

In order to put these results in perspective, we note that the Netflix competition (which carried a 1 million dollar prize) was about improving the RMSE compared to the recommender system of Netflix by 10%.[1]

In terms of multiplicative relative performance, for some categories the accuracy of SmarterDeals is much better than ACF's (e.g., for Automotive our approach outperforms ACF by 173.4%, and 37.5% on average).

There are two parent categories, Professional Services and Public Services and Government, for which our method did not do very well. These two parent categories seem very generic and they are not properly categorized in specific sub-categories. They cover a wide variety of services and it might be the reason for negative results for the relative improvement of our approach with respect to ACF approach. If we could divide these services into more specific categories, there is the possibility that our method outperforms the ACF method.

[1]The last years of the competition saw most of the teams improving by less than 2%.

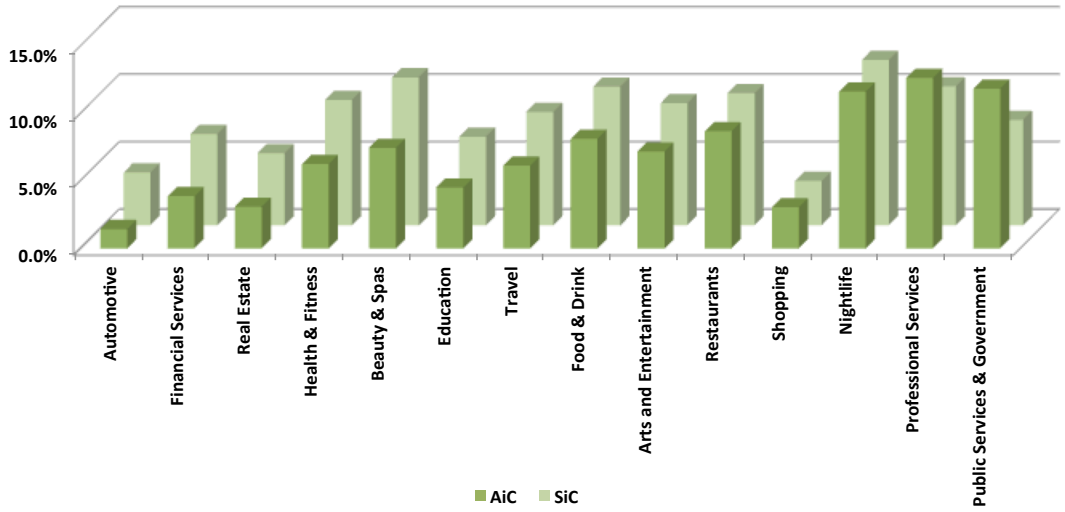

Figure 5.1: Improvement of ACF (AiC), and SmarterDeals (SiC) with respect to classic CF (C)

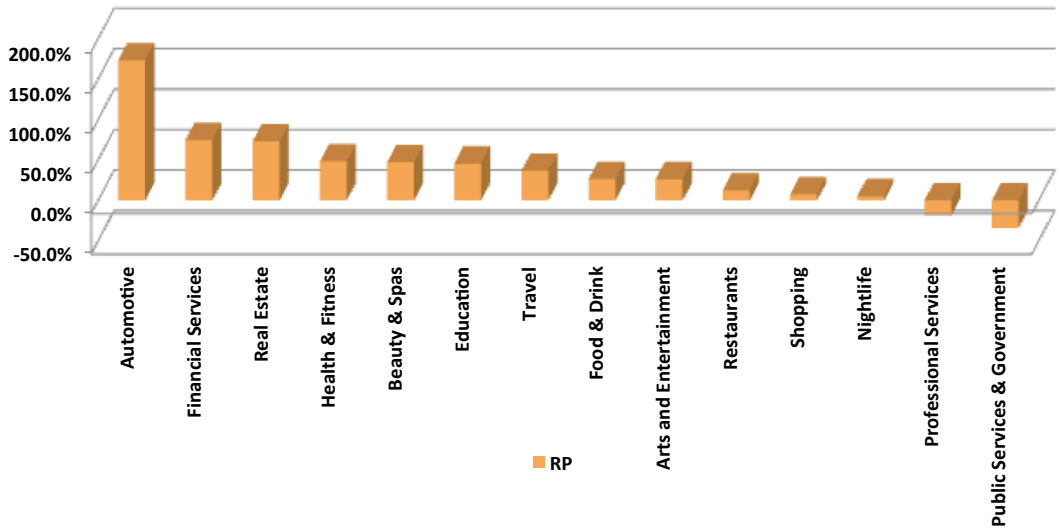

Figure 5.2: Relative performance of SmarterDeals (S) with respect to ACF (A)

Chapter 6

Conclusions

Recommender systems are software applications and tools that provide recommendations to users for different products and services. The recommendation techniques are based on user preferences and profiles to suggest relevant recommendations for each user. By providing personalized recommendations, recommender systems help users find the products and services which may be more interesting for them with less effort. Although, due to the overload and fast growth of digital information and technology in today's world, there is still significant need and space for improvements and optimizations in recommendation techniques. Moreover, there are many more application areas that can benefit from recommender systems. An important part of these improvement techniques involves incorporating contextual information in recommender systems.

This research proposed a context-aware recommendation technique which is developed as a deal recommendation system. A large number of users and the popularity of Groupon, a daily deal website, made us think about developing a deal recommendation system using Groupon as a resource for interesting deals. To implement our recommender system, we applied one of the most advanced recommendation tech-

niques introduced by Bell and Koren, winners of the grand Netflix prize. However, the most important contribution of our system is context-awareness. We took advantage of SmarterContext, a dynamic context management system, to gather, evaluate, and maintain contextual information for our system. Finally, by incorporating contextual information in our recommendation engine and evaluating the system using Yelp's Academic Dataset, we demonstrated that our context-aware recommendation system outperforms both the traditional collaborative filtering technique and Bell and Koren's advanced collaborative filtering approach in most cases.

Limitations in public data sets containing real user contextual information results in constraints for studying and experimenting with our system according to other interesting areas of users' contextual information (e.g., their social relationships). Since SmarterContext is designed to be able to provide a large variety of users' context information, we are interested in developing and evaluating our system for other possible contextual areas in the future.

Appendix A

Additional Information

The Resource Description Framework (RDF) is a general method for conceptual description or modelling of information in web resources [49]. The following RDF file presents a sample of user's contextual information which has been modelled with the RDF method using the SmarterContext vocabulary.

sources/user4.rdf

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:pwc="http://smartercontext.org/vocabularies/pwc/v1.1/pwc.owl#"
    xmlns:gc="http://smartercontext.org/vocabularies/gc/v1.1/gc.owl#"
    xmlns:shopping="http://smartercontext.org/vocabularies/shopping/v1
        .1/shopping.owl#" >
  <rdf:Description rdf:about="http://smartercontext.org/vocabularies/rdf
      /user4.rdf#user4">
    <rdf:type rdf:resource="http://smartercontext.org/vocabularies/pwc/
        v1.1/pwc.owl#User"/>
    <pwc:givenName rdf:datatype="http://www.w3.org/2001/XMLSchema#string
        ">Nicholas M.</pwc:givenName>
    <pwc:preferredLocation rdf:resource="http://smartercontext.org/
        vocabularies/rdf/geo.rdf#Berkeley"/>
```

```
    <pwc:ranked rdf:resource="http://smartercontext.org/vocabularies/rdf
        /dealcategories.owl#Fitness &amp;amp; Instruction"/>
    <pwc:ranked rdf:resource="http://smartercontext.org/vocabularies/rdf
        /dealcategories.owl#Desserts"/>
    <pwc:ranked rdf:resource="http://smartercontext.org/vocabularies/rdf
        /dealcategories.owl#Active Life"/>
    <pwc:ranked rdf:resource="http://smartercontext.org/vocabularies/rdf
        /dealcategories.owl#Coffee &amp;amp; Tea"/>
    <pwc:ranked rdf:resource="http://smartercontext.org/vocabularies/rdf
        /dealcategories.owl#Gyms"/>
    <pwc:ranked rdf:resource="http://smartercontext.org/vocabularies/rdf
        /dealcategories.owl#Amateur Sports Teams"/>
    <pwc:ranked rdf:resource="http://smartercontext.org/vocabularies/rdf
        /dealcategories.owl#Food"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://smartercontext.org/vocabularies/rdf
      /geo.rdf#Berkeley">
    <rdf:type rdf:resource="http://smartercontext.org/vocabularies/gc/v1
        .1/gc.owl#GeoLocation"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://smartercontext.org/vocabularies/rdf
      /dealcategories.owl#Amateur Sports Teams">
    <pwc:rankingValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int
        ">5.0</pwc:rankingValue>
    <rdf:type rdf:resource="http://smartercontext.org/vocabularies/
        shopping/v1.1/shopping.owl#ProductServiceCategory"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://smartercontext.org/vocabularies/rdf
      /dealcategories.owl#Gyms">
    <pwc:rankingValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int
        ">5.0</pwc:rankingValue>
```

```
  <rdf:type rdf:resource="http://smartercontext.org/vocabularies/
      shopping/v1.1/shopping.owl#ProductServiceCategory"/>
</rdf:Description>
<rdf:Description rdf:about="http://smartercontext.org/vocabularies/rdf
    /dealcategories.owl#Coffee &amp;amp; Tea">
  <pwc:rankingValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int
      ">3.0</pwc:rankingValue>
  <rdf:type rdf:resource="http://smartercontext.org/vocabularies/
      shopping/v1.1/shopping.owl#ProductServiceCategory"/>
</rdf:Description>
<rdf:Description rdf:about="http://smartercontext.org/vocabularies/rdf
    /dealcategories.owl#Desserts">
  <pwc:rankingValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int
      ">2.0</pwc:rankingValue>
  <rdf:type rdf:resource="http://smartercontext.org/vocabularies/
      shopping/v1.1/shopping.owl#ProductServiceCategory"/>
</rdf:Description>
<rdf:Description rdf:about="http://smartercontext.org/vocabularies/rdf
    /dealcategories.owl#Active Life">
  <pwc:rankingValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int
      ">5.0</pwc:rankingValue>
  <rdf:type rdf:resource="http://smartercontext.org/vocabularies/
      shopping/v1.1/shopping.owl#ProductServiceCategory"/>
</rdf:Description>
<rdf:Description rdf:about="http://smartercontext.org/vocabularies/rdf
    /dealcategories.owl#Fitness &amp;amp; Instruction">
  <pwc:rankingValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int
      ">5.0</pwc:rankingValue>
  <rdf:type rdf:resource="http://smartercontext.org/vocabularies/
      shopping/v1.1/shopping.owl#ProductServiceCategory"/>
</rdf:Description>
```

```
  <rdf:Description rdf:about="http://smartercontext.org/vocabularies/rdf
      /dealcategories.owl#Food">
    <pwc:rankingValue rdf:datatype="http://www.w3.org/2001/XMLSchema#int
        ">3.0</pwc:rankingValue>
    <rdf:type rdf:resource="http://smartercontext.org/vocabularies/
        shopping/v1.1/shopping.owl#ProductServiceCategory"/>
  </rdf:Description>
</rdf:RDF>
```

Bibliography

[1] S. Abbar, M. Bouzeghoub, and S. Lopez. Context-Aware Recommender Systems: A Service-Oriented Approach. In *VLDB*, 2009.

[2] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *Proceedings 1st International Symposium on Handheld and Ubiquitous Computing*, (HUC '99), pages 304–307, London, UK, UK, 1999. Springer-Verlag.

[3] Gediminas Adomavicius, Ramesh Sankaranarayanan, Shahana Sen, and Alexander Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems*, 23(1):103–145, January 2005.

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

[5] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, (RecSys 2008), pages 335–336, New York, NY, USA, 2008. ACM.

[6] Sarabjot Singh Anand and Bamshad Mobasher. Contextual recommendation. In Bettina Berendt, Andreas Hotho, Dunja Mladenic, and Giovanni Semeraro, editors, *From Web to Social Web: Discovering and Deploying User and Content Profiles*, pages 142–160. Springer-Verlag, Berlin, Heidelberg, 2007.

[7] Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[8] Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, March 1997.

[9] Linas Baltrunas. Exploiting contextual information in recommender systems. In *Proceedings of the 2008 ACM conference on Recommender systems*, (RecSys '08), pages 295–298, New York, NY, USA, 2008. ACM.

[10] Linas Baltrunas and Xavier Amatriain. Towards Time-Dependant Recommendation based on Implicit Feedback. In *Context-aware Recommender Systems Workshop at Recsys 2009*, 2009.

[11] Linas Baltrunas and Francesco Ricci. Context-based splitting of item ratings in collaborative filtering. In *Proceedings of the Third ACM Conference on Recommender Systems*, (RecSys 2009), pages 245–248, New York, NY, USA, 2009. ACM.

[12] Chumki Basu, Haym Hirsh, and William Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *In Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 714–720. AAAI Press, 1998.

[13] Mary Bazire and Patrick Brézillon. Understanding context before using it. In *Proceedings of the 5th international conference on Modeling and Using Context*, (CONTEXT '05), pages 29–40, Berlin, Heidelberg, 2005. Springer-Verlag.

[14] Nicholas J. Belkin and W. Bruce Croft. Information filtering and information retrieval: two sides of the same coin? *Commun. ACM*, 35(12):29–38, December 1992.

[15] Claudio Bettini, Oliver Brdiczka, Karen Henricksen, Jadwiga Indulska, Daniela Nicklas, Anand Ranganathan, and Daniele Riboni. A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing*, 6(2):161–180, April 2010.

[16] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data - the story so far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.

[17] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithm for collaborative filtering. In *Proceedings of the 14 th Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

[18] Surajit Chaudhuri and Umeshwar Dayal. An overview of data warehousing and olap technology. *SIGMOD Record*, 26(1):65–74, March 1997.

[19] Annie Chen. Context-aware collaborative filtering system: predicting the user's preferences in ubiquitous computing. In *CHI '05 extended abstracts on Human factors in computing systems*, (CHI EA 2005), pages 1110–1111, New York, NY, USA, 2005. ACM.

[20] Mark Claypool, Anuja Gokhale, Tim Miranda, Pavel Murnikov, Dmitry Netes, and Matthew Sartin. Combining content-based and collaborative filters in an

online newspaper. In *In Proceedings of ACM SIGIR Workshop on Recommender Systems*, 1999.

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

[22] Econsultancy - Digital Marketers United. Groupon: your email marketing is failing your fans. http://econsultancy.com/us/blog/7291- groupon-your-email-marketing-is-failing- your-audience, April 2011.

[23] Bracha Shapira Francesco Ricci, Lior Rokach. Introduction to recommender systems handbook. *Springer US*, October 2010.

[24] Lilien GL, Kotler P, and Moorthy KS. *Marketing models*. Prentice Hall, 1992.

[25] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, December 1992.

[26] Google Inc. The Google Product Taxonomy. http://support.google.com/merchants/bin/answer.py?hl=en&answer=160081, 2012.

[27] Groupon. Groupon API v2. https://sites.google.com/site/grouponapiv2, 2012.

[28] Scott Guinn, Simon Ellis, Michael Fauscette, Leslie Hand, Mary Wardley, and Kimberly Knickle. IBM synchronizes its commerce 2.0 strategy with smarter commerce initiative. Technical report, IBM Corporation, 2011.

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

[30] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '95, pages 194–201, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[31] Jong-yi Hong, Eui-ho Suh, and Sung-Jin Kim. Context-aware systems: A literature review and classification. *Expert Syst. Appl.*, 36(4):8509–8522, May 2009.

[32] Yelp Inc. Yelp's academic dataset. http://www.yelp.com/academic-dataset, 2012.

[33] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, (RecSys '10), pages 79–86, New York, NY, USA, 2010. ACM.

[34] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl. Grouplens: applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87, March 1997.

[35] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, (KDD '08), pages 426–434, New York, NY, USA, 2008. ACM.

[36] Yehuda Koren and Robert Bell. Advances in collaborative filtering. *Recommender Systems Handbook, Springer US*, pages 145–186, 2011.

[37] Cosimo Palmisano, Alexander Tuzhilin, and Michele Gorgoglione. Using context to improve predictive modeling of customers in personalization applications. *IEEE Transactions on Knowledge and Data Engineering*, 20(11):1535–1549, November 2008.

[38] Umberto Panniello, Alexander Tuzhilin, Michele Gorgoglione, Cosimo Palmisano, and Anto Pedone. Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems. In *Proceedings of the third ACM conference on Recommender systems*, (RecSys '09), pages 265–268, New York, NY, USA, 2009. ACM.

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

[40] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, (SIGIR '11), pages 635–644, New York, NY, USA, 2011. ACM.

[41] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, (CSCW '94), pages 175–186, New York, NY, USA, 1994. ACM.

[42] Elaine Rich. User modeling via stereotypes. *COGNITIVE SCIENCE 3*, 1979.

[43] Gerard Salton. *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[44] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, (WWW '01), pages 285–295, New York, NY, USA, 2001. ACM.

[45] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, (SIGIR '02), pages 253–260, New York, NY, USA, 2002. ACM.

[46] Upendra Shardanand and Pattie Maes. Social information filtering: algorithms for automating ``word of mouth''. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, (CHI '95), pages 210–217, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[47] The Economist. Groupon Anxiety. http://www.economist.com/node/18388904, March 2011.

[48] The World Wide Web Consortium (W3C). OWL Web Ontology Language Reference. http://www.w3.org/TR/owl-ref/, 2004.

[49] The World Wide Web Consortium (W3C). RDF Vocabulary Description Language 1.0: RDF Schema. http://www.w3.org/TR/rdf-schema/, 2004.

[50] N. M. Villegas and H. A. Müller. The smartercontext ontology and its application to the smart internet: A smarter commerce case study. *Second Book on the Personal Web, LNCS. Springer*, 2012.

[51] Norha M. Villegas and Hausi A. Müller. Managing dynamic context to optimize smart interactions and services. In Mark Chignell, James Cordy, Joanna Ng, and Yelena Yesha, editors, *The Smart Internet: Current Research and Future Applications*, chapter Managing dynamic context to optimize smart interactions and services, pages 289–318. Springer-Verlag, Berlin, Heidelberg, 2010.

[52] Norha M. Villegas, Hausi A. Müller, Juan C. Muñoz, Alex Lau, Joanna Ng, and Chris Brealey. A dynamic context management infrastructure for supporting user-driven web integration in the personal web. In *Proceedings of the 2011 Conference of the Center for Advanced Studies on Collaborative Research*, (CASCON '11), pages 200–214, Riverton, NJ, USA, 2011. IBM Corp.

[53] Will Wade. A grocery cart that holds bread, butter and preferences. http://www.nytimes.com/2003/01/16/technology/circuits/16safe.html, 2003.