Decisions@RecSys

Proceedings of the

RecSys 2013

Workshop on

Human Decision Making in Recommender Systems

(Decisions@RecSys'13)

October 12, 2013

In conjunction with the

7th ACM Conference on Recommender Systems

October 12-16, 2013, Hong Kong, China

Preface

Users interact with recommender systems to obtain useful information about products or services that may be of interest for them. But, while users are interacting with a recommender system to fulfill a primary task, which is usually the selection of one or more items, they are facing several other decision problems. For instance, they may be requested to select specific feature values (e.g., camera's size, zoom) as criteria for a search, or they could have to identify features to be used in a critiquing based recommendation session, or they may need to select a repair proposal for inconsistent user preferences when interacting with a recommender. In all these scenarios, and in many others, users of recommender systems are facing decision tasks.

The complexity of decision tasks, limited cognitive resources of users, and the tendency to keep the overall decision effort as low as possible is modeled by theories that conjecture "bounded rationality", i.e., users are exploiting decision heuristics rather than trying to take an optimal. Furthermore, preferences of users will likely change throughout a recommendation session, i.e., preferences are constructed in a specific decision context and users may not fully know their preferences beforehand. Within the scope of a decision process, preferences are strongly influenced by the goals of the customer, existing cognitive constraints, and the personal experience of the customer. Due to the fact that users do not have stable preferences, the interaction mechanisms provided by a recommender system and the information shown to a user can have an enormous impact on the outcome of a decision process.

Theories from decision psychology and cognitive psychology have already elaborated a number of methodological tools for explaining and predicting the user behavior in these scenarios. The major goal of this workshop is to establish a platform for industry and academia to present and discuss new ideas and research results that are related to the topic of human decision making in recommender systems. The workshop consists of a mix of six presentations of papers in which results of ongoing research as reported in these proceedings are presented and two invited talks: Bart Knijnenburg presenting "Simplifying privacy decisions: towards interactive and adaptive solutions" and and Jill Freyne and Shlomo Berkovsky presenting: "Food Recommendations: Biases that Underpin Ratings". The workshop is closed by a final discussion session.

Li Chen, Marco de Gemmis, Alexander Felfernig, Pasquale Lops,
Francesco Ricci, Giovanni Semeraro and Martijn Willemsen
September 2013

Workshop Committee

Workshop Co-Chairs

Li Chen, Hong Kong Baptist University
Marco de Gemmis, University of Bari Aldo Moro, Italy
Alexander Felfernig, Graz University of Technology, Austria
Pasquale Lops, University of Bari Aldo Moro, Italy
Francesco Ricci, University of Bozen-Bolzano, Italy
Giovanni Semeraro, University of Bari Aldo Moro, Italy
Martijn Willemsen, Eindhoven University of Technology, Netherlands

Organization

Gerald Ninaus, Graz University of Technology

Program Committee

David Amid, IBM Haifa Research Center
Shlomo Berkovsky, NICTA
Robin Burke, DePaul University
Li Chen, Hong Kong Baptist University
Marco De Gemmis, Dipartimento di Informatica – University of Bari
Alexander Felfernig, Graz University of Technology
Gerhard Friedrich, Alpen-Adria-Universitaet Klagenfurt
Sergiu Gordea, AIT
Anthony Jameson, DFKI
Dietmar Jannach, TU Dortmund
Bart Knijnenburg, University of California, Irvine
Gerhard Leitner, Alpen-Adria-Universitaet Klagenfurt
Pasquale Lops, University of Bari
Gerald Ninaus, Graz University of Technology
Florian Reinfrank, Graz University of Technology
Francesco Ricci, Free University of Bozen-Bolzano
Giovanni Semeraro, Dipartimento di Informatica – University of Bari
Ofer Shir, IBM Research
Erich Teppan, Alpen-Adria-Universitaet Klagenfurt
Martijn Willemsen, Eindhoven University of Technology
Markus Zanker, Alpen-Adria-Universitaet Klagenfurt

Table of Contents

Efficiency Improvement of Neutrality-Enhanced Recommendation

Toshihiro Kamishima, Shotaro Akaho, and Hideki Asoh
National Institute of Advanced Industrial Science and Technology (AIST)
AIST Tsukuba Central 2, Umezono 1-1-1, Tsukuba, Ibaraki, 305-8568 Japan
mail@kamishima.net, s.akaho@aist.go.jp, h.asoh@aist.go.jp

Jun Sakuma
University of Tsukuba
1-1-1 Tennodai, Tsukuba, 305-8577 Japan
jun@cs.tsukuba.ac.jp

ABSTRACT

This paper proposes an algorithm for making recommendations so that neutrality from a viewpoint specified by the user is enhanced. This algorithm is useful for avoiding decisions based on biased information. Such a problem is pointed out as the filter bubble, which is the influence in social decisions biased by personalization technologies. To provide a neutrality-enhanced recommendation, we must first assume that a user can specify a particular viewpoint from which the neutrality can be applied, because a recommendation that is neutral from all viewpoints is no longer a recommendation. Given such a target viewpoint, we implement an information-neutral recommendation algorithm by introducing a penalty term to enforce statistical independence between the target viewpoint and a rating. We empirically show that our algorithm enhances the independence from the specified viewpoint.

Keywords

recommender system, neutrality, fairness, filter bubble, collaborative filtering, matrix factorization, information theory

1. INTRODUCTION

A recommender system searches for items or information that is estimated to be useful to a user based on the user's prior behaviors and the features of items. Over the past decade, such recommender systems have been introduced and managed at many e-commerce sites to promote items sold at those sites. The influence of personalization technologies such as recommender systems or personalized search engines on people's decision making is considerable. For example, at a shopping site, if a customer checks a recommendation list and finds five-star-rated items, he/she will more seriously consider buying these strongly recommended items. These technologies have thus become an indispensable tool for users. However, the problem of *filter bubble*, which is the unintentional bias or the limited diversity of information provided to users, has accompanied the growing influence of personalization algorithms.

The term filter bubble was recently coined by Pariser [12]. Due to the strong influence of personalized technologies, the topics of information provided to users are becoming restricted to those originally preferred by them, and this restriction is not perceived by users. In this way, each individual is metaphorically enclosed in his/her own separate *bubble*. Pariser claimed that users lose the opportunity to find new interests because of the limitations of the bubbles created around their original interests, and that sharing reasonable yet opposing viewpoints on public issues affecting our society is thus becoming more difficult. To discuss this filter bubble problem, a panel discussion was held at the RecSys 2011 conference [14].

During the RecSys panel discussion, panelists made the following assertions about the filter bubble problem. The diversity of topics is certainly biased by the influence of personalization. At the same time, it is impossible to make recommendations that are absolutely neutral from any viewpoint, and thus there is a trade-off between focusing on topics that better fit users' interests or needs and enhancing the varieties of provided topics. To address this problem, the panelists also pointed out several possible directions: taking into account users' immediate needs as well as their long-term needs; optimizing a recommendation list as a whole; and providing tools for perspective-taking.

To our knowledge, there is no major tool that enables users to control their perspective to address this filter bubble problem. We therefore advocate a new *information-neutral recommender system* that guarantees the neutrality of recommendations. As pointed out during the RecSys 2011 panel discussion, it is impossible to make a recommendation that is absolutely neutral from all viewpoints, and we therefore focus on neutrality from a viewpoint or type of information specified by the user. For example, users can specify a feature of an item, such as a brand, or a user feature, such as a gender or an age, as a viewpoint. An information-neutral recommender system is designed so that these specified features will not influence the recommendation results. This system can also be used to ensure fair treatment of content providers or product suppliers or to avoid the use of infor-

RecSys'13, October 12–16, 2013, Hong Kong, China.
Paper presented at the 2013 Decisions@RecSys workshop in conjunction with the 7th ACM conference on Recommender Systems.

mation that is restricted by law or regulation.

Last year at this Decisions workshop, we borrowed the idea of fairness-aware data mining, which we had proposed earlier [8], to build an information-neutral recommender system of the type described above [7]. To enhance neutrality or independence in recommendations, we introduced a constraint term that represents the mutual information between a recommendation result and a specified viewpoint. The naive implementation of this constraint term did indeed enhance the neutrality of recommendations, but there remained serious shortcomings in its scalability. In this paper, therefore, we advocate several new formulations of this constraint term that are more scalable.

Our contributions are as follows. First, we present a definition of neutrality in recommendation based on the consideration of why it is impossible to achieve an absolutely neutral recommendation. Second, we propose a method to enhance the neutrality of a probabilistic matrix factorization model. Finally, we demonstrate that the neutrality of a recommender system can be enhanced.

In section 2, we discuss the filter bubble problem and the concept of neutrality in recommendation, and define the goal of an information-neutral recommendation task. An information-neutral recommender system is proposed in section 3, and the experimental results of its application are shown in section 4. Sections 5 and 6 cover related work and our conclusion, respectively.

2. INFORMATION NEUTRALITY

In this section, we discuss information neutrality in recommendation based on an examination of the filter bubble problem and the ugly duckling theorem.

2.1 The Filter Bubble Problem

We will first summarize the filter bubble problem posed by Pariser and the panel discussion about this problem held at the RecSys 2011 conference. The *Filter Bubble* problem is the concern that personalization technologies narrow and bias the topics of information provided to people, who do not notice this phenomenon [12].

Pariser demonstrated the following examples in a TED talk about this problem [11]. Users of the social network service Facebook specify other users as *friends* with whom they then can chat, have private discussions, and share information. To help users find their friends, Facebook provides a recommendation list of others who are expected to be related to a user. When Pariser started to use Facebook, the system showed a friend recommendation list that consisted of both conservative and progressive people. However, because he more frequently selected progressive people as friends, conservative people were increasingly excluded from his recommendation list by a personalization functionality. Pariser claimed that, in this way, the system excluded conservative people without his permission and that he lost the opportunity to be exposed to a wide variety of opinions.

Pariser's claims can be summarized as follows. First, personalization technologies restrict an individual's opportunities to obtain information about a wide variety of topics. The chance to gain knowledge that could ultimately enhance an individual's life is lessened. Second, the individual obtains information that is too personalized; thus, the amount of shared information and shared debate in our society is decreased. Pariser asserts that the loss of shared information is a serious obstacle for building social consensus. He claimed that the personalization of information thereby becomes a serious obstacle for building consensus.

RecSys 2011 featured a panel discussion on this filter bubble problem [14]. The panel concentrated on the following three points: (a) Are there filter bubbles? (b) To what degree is personalized filtering a problem? and (c) What should we as a community do to address the filter bubble problem? Among these points, we focus on the point (c). The panelists presented several directions to explore in addressing the filter bubble problem. First, a system could consider users' immediate needs as well as their long-term needs. Second, instead of selecting individual items separately, a recommendation list or portfolio could be optimized as a whole. And Finally a system could provide tools for perspective-taking to see the world through other viewpoints.

2.2 Neutrality in Recommendation

Among the directions for addressing the filter bubble, we here take the approach of providing a tool for perspective-taking. Before presenting this tool, we explored the notion of neutrality based on the ugly duckling theorem. The *ugly duckling theorem* is a classical theorem in pattern recognition literature that asserts the impossibility of classification without weighing certain features or aspects of objects as more important than others [17]. Consider a case in which 2^n ducklings are represented by n binary features and are classified into positive or negative classes based on these features. It is easy to show that the number of possible decision rules based on these features to discriminate an ugly duckling and a normal duckling is equal to the number of patterns to discriminate any pair of normal ducklings. In other words, every duckling resembles a normal duckling and an ugly duckling equally. This counterintuitive conclusion is deduced from the premise that all features are treated equally. Attention to an arbitrary feature such as black feathers makes an ugly duckling ugly. When we classify something, we of necessity weigh certain features, aspects, or viewpoints of classified objects. Because recommendation is considered a task for classifying whether items are interesting or not, certain features or viewpoints inevitably must be weighed when making a recommendation. Consequently, the absolutely neutral recommendation is impossible, as pointed out in the RecSys panel.

We propose a neutral recommendation framework other than the absolutely neutral recommendation. Recalling the ugly duckling theorem, we must focus on certain features or viewpoints in classification. This fact indicates that it is feasible to make a recommendation that is neutral from a specific viewpoint instead of all viewpoints. We hence advocate an *information-neutral recommender system* (INRS) that enhances the neutrality in recommendation from the viewpoint specified by a user. In Pariser's Facebook example, a system could enhance the neutrality so that recommended friends are both conservative and progressive, but the system would be allowed to make biased decisions in terms of the other viewpoints, e.g., the birthplace or age of friends.

We formally model this neutrality by the statistical independence between recommendation results and viewpoint values, i.e., $\Pr[R|V] = \Pr[R]$. This means that the same recommendations are made for the cases where all conditions are the same except for the viewpoint values. In other words, no information of viewpoint features influences the recommendation results according to the information theory. An INRS hence tends to be less accurate, because useable information is decreased. In the example of a friend recommendation, no matter what a user's political conviction is, the conviction is ignored and excluded in the process of making a recommendation.

We wish to emphasize that neutrality is distinct from recommendation diversity, which is the attempt to recommend items that are mutually less similar. Topic diversification is one of the proposed techniques for enhancing diversity by excluding similar items from a recommendation list [20]. The constraint term in [19] is designed to exclude similar items from a final list. Therefore, while neutrality involves the relation between recommendations and single viewpoint features, diversity concerns the mutual relation among recommendations. Inversely, enhancing the diversity cannot suppress the use of specific information, and an INRS is allowed to offer mutually similar items. In the case of the friend recommendation, if a progressive person is recommended as a friend, the INRS will recommend another person whose conditions other than political convictions are the same. In the case of the diversified recommendation, one of two persons would not be recommended because the two persons are very similar.

The INRS is beneficial not only for users but also for system managers. It can be used to ensure the fair treatment of content providers or product suppliers. The federal trade commission has been investigating Google to determine whether the search engine ranks its own services higher than those of competitors [3]. E-commerce sites want to treat their product suppliers fairly when making recommendations to their customers. If a brand of providers or suppliers is specified as a viewpoint, a system can make recommendations that are neutral in terms of the items' brands. An information-neutral recommendation is also helpful for avoiding the use of information that is restricted by law or regulation. For example, the use of some information is prohibited for the purpose of making recommendations by privacy policies. In this case, by treating the prohibited information as a viewpoint, recommendations can be neutral in terms of the prohibited information.

3. THE INFORMATION-NEUTRAL RECOMMENDER SYSTEM

We formalize the task of information-neutral recommendation and present an algorithm for performing this task.

3.1 Task Formalization

Recommendation tasks can be classified into three types: *recommending good items* that meet a user's interest, *optimizing the utility* of users, and *predicting item ratings* of items for a user [5]. Among these tasks, we here concentrate on the task of predicting ratings. $X \in \{1, \ldots, n\}$ and $Y \in \{1, \ldots, m\}$ denote random variables for the user and item, respectively. An event (x, y) is an instance of a pair (X, Y). R denotes a random variable for the rating of Y as given by X, and its instance is denoted by r. We here assume that the domain of ratings is the set of real values. These variables are in common with an original predicting ratings task.

To enhance information neutrality in recommendation, we additionally introduced a viewpoint random variable, V, which indicates the viewpoint feature from which the neutrality is enhanced. This variable is specified by a user, and its value depends on various aspects of an event. Possible examples of viewpoint variables are a user's gender, which is part of the user component of an event, a movie's release year, which is part of the item component of an event, and the timestamp when a user rates an item, which would belong to both elements in an event. In this paper, we restrict the domain of a viewpoint variable to a binary type, $\{0, 1\}$, for simplicity. A training sample consists of an event, (x, y), a viewpoint value for the event, v, and a rating value for the event, r. A training set is a set of N training samples, $\mathcal{D} = \{(x_i, y_i, v_i, r_i)\}$, $i = 1, \ldots, N$.

Given a new event, (x, y), and its corresponding viewpoint value, v, a rating prediction function, $\hat{r}(x, y, v)$, predicts a rating of the item y by the user x, and satisfies $\hat{r}(x, y, v) = \mathbb{E}_{\Pr[R|x,y,v]}[R]$. This rating prediction function is estimated by optimizing an objective function having three components: a loss function, $\mathrm{loss}(r^*, \hat{r})$, a neutrality term, $\mathrm{neutral}(R, V)$, and a regularization term, reg. The loss function represents the dissimilarity between a true rating value, r^*, and a predicted rating value, $\hat{r}$. The neutrality term quantifies the expected degree of neutrality of the predicted rating values from a viewpoint expressed by a viewpoint feature, and its larger value indicates the higher level of neutrality. The aim of the regularization term is to avoid over-fitting. Given a training sample set, $\mathcal{D}$, the goal of the information-neutral recommendation (predicting rating case) is to acquire a rating prediction function, $\hat{r}(x, y, v)$, so that the expected value of the loss function is as small as possible and the neutral term is as large as possible. We formulate this goal by finding a rating prediction function, $\hat{r}$, so as to minimize the following objective function:

$$\sum_{\mathcal{D}} \mathrm{loss}(r, \hat{r}(x, y, v)) + \eta\, \mathrm{neutral}(R, V) + \lambda\, \mathrm{reg}(\mathbf{\Theta}), \qquad (1)$$

where $\eta > 0$ is a neutrality parameter to balance between the loss and the neutrality, $\lambda > 0$ is a regularization parameter, and $\mathbf{\Theta}$ is a set of model parameters.

3.2 Probabilistic Matrix Factorization Model

In this paper, we adopt a probabilistic matrix factorization model [15] to predict ratings, because this model is highly effective in its prediction accuracy as well as efficient in its scalability. Though there are several minor variants of this model, we here use the following model defined as equation (3) in [9]:

$$\hat{r}(x, y) = \mu + b_x + c_y + \mathbf{p}_x^\top \mathbf{q}_y, \qquad (2)$$

where μ, b_x, and c_y are global, per-user, and per-item bias parameters, respectively, and $\mathbf{p}_x$ and $\mathbf{q}_y$ are K-dimensional parameter vectors, which represent the cross effects between users and items. We then adopt the following squared loss with a regularization term:

$$\sum_{(x_i, y_i, r_i) \in \mathcal{D}} (r_i - \hat{r}(x_i, y_i))^2 + \lambda\, \mathrm{reg}(\mathbf{\Theta}). \qquad (3)$$

This model is proved to be equivalent to assuming that true rating values are generated from a normal distribution whose mean is equation (2). If all samples over all X and Y are available, the objective function is convex; and thereby globally optimal parameters can be derived by a simple gradient descent method. Unfortunately, because not all samples are observed, the loss function (3) is non-convex, and only local optima can be found. However, it is empirically known that a simple gradient method succeeds in finding a good solution in most cases [9].

We then extend this model to enhance the information neutrality. First, we modify the model of equation (2) so that it is dependent on the viewpoint value, v. For each value of V, 0 and 1, we prepare a parameter set, $\mu^{(v)}$, $b_x^{(v)}$, $c_y^{(v)}$, $\mathbf{p}_x^{(v)}$, and $\mathbf{q}_y^{(v)}$. One of the parameter sets is chosen according to the viewpoint value, and we get the rating prediction function:

$$\hat{r}(x,y,v) = \mu^{(v)} + b_x^{(v)} + c_y^{(v)} + \mathbf{p}_x^{(v)\top}\mathbf{q}_y^{(v)}. \tag{4}$$

By substituting equations (4) into equation (1) and adopting a squared loss function as in the original probabilistic matrix factorization case, we obtain an objective function of an information-neutral recommendation model:

$$\sum_{(x_i,y_i,r_i,v_i)\in\mathcal{D}} (r_i - \hat{r}(x_i,y_i,v_i))^2 + \eta\,\mathrm{neutral}(R,V) + \lambda\,\mathrm{reg}(\mathbf{\Theta}), \tag{5}$$

where the regularization term is a sum of L_2 regularizers of parameter sets for each value of v except for global biases, $\mu^{(v)}$. Model parameters, $\mathbf{\Theta}^{(v)} = \{\mu^{(v)}, b_x^{(v)}, c_y^{(v)}, \mathbf{p}_x^{(v)}, \mathbf{q}_y^{(v)}\}$, for $v \in \{0,1\}$, are estimated so as to minimize this objective. Once we learn the parameters of the rating prediction function, we can predict a rating value for any event by applying equation (4).

3.3 Neutrality Term

Now, all that remains is to define a neutrality term. As described in section 2.2, we formalize the neutrality as the statistical independence between a recommendation result and a viewpoint feature. We propose neutrality terms that are based on mutual information and Calders-Verwer's discrimination score, both of which quantify the degree of independence between R and V.

3.3.1 Mutual Information

We first use the same idea as in [8] and quantify the degree of the neutrality by negative mutual information under the assumption that neutrality can be regarded as statistical independence. Negative mutual information between R and V is defined as:

$$\begin{aligned}
-\mathrm{I}(R;V) &= -\sum_V \int \Pr[R,V] \log \frac{\Pr[R|V]}{\Pr[R]} dR \\
&\approx -\frac{1}{N} \sum_{(x_i,y_i,v_i)\in\mathcal{D}} \log \frac{\Pr[\hat{r}_i|v_i]}{\Pr[\hat{r}_i]} \\
&= -\frac{1}{N} \sum_{(x_i,y_i,v_i)\in\mathcal{D}} \log \frac{\Pr[\hat{r}_i|v_i]}{\sum_{v\in\{0,1\}} \Pr[\hat{r}_i|v]\Pr[v]},
\end{aligned} \tag{6}$$

where $\hat{r}_i$ is derived by applying $(x_i,y_i,v_i) \in \mathcal{D}$ to equation (4). The marginalization over R and V is approximated by the sample mean over $\mathcal{D}$ in the second line, and we use a sample mass function as $\Pr[V]$. $\Pr[R|V]$ can be derived by marginalizing $\Pr[R|X,Y,V]\Pr[X,Y]$ over X and Y. We again approximate this marginalization by the sample mean and get:

$$\Pr[r|v] \approx \frac{1}{|\mathcal{D}^{(v)}|} \sum_{(x_i,y_i)\in\mathcal{D}^{(v)}} \mathrm{Normal}\left(r;\, \hat{r}(x_i,y_i,v), \mathbb{V}_{\mathcal{D}^{(v)}}(R)\right), \tag{7}$$

where $\mathrm{Normal}(\cdot)$ is a pdf of normal distribution, $\mathcal{D}^{(v)}$ consists of all training samples whose viewpoint values are equal to v, and $\mathbb{V}_{\mathcal{D}^{(v)}}(R)$ is a sample variance,

$$\tfrac{1}{|\mathcal{D}^{(v)}|} \textstyle\sum_{r_i\in\mathcal{D}^{(v)}} (r_i - \mathbb{M}_{\mathcal{D}^{(v)}}(\{\hat{r}\}))^2,$$

where $\mathbb{M}_{\mathcal{D}}(\{\hat{r}\})$ is

$$\mathbb{M}_{\mathcal{D}}(\{\hat{r}\}) = \tfrac{1}{|\mathcal{D}|} \textstyle\sum_{(x_i,y_i,v_i)\in\mathcal{D}} \hat{r}(x_i,y_i,v_i).$$

This is very hard to manipulate because this is a mixture distribution with an enormous number of components. We hence took an approach of directly modeling $\Pr[r|v]$, and used two types of models.

The first one is a histogram model, which was proposed in our preliminary work [7]. Though rating values are treated as real values, they are originally discrete scores. Therefore, a set of predicted ratings, $\{\hat{r}_i\}$, are divided into bins. Given a set of intervals, $\{\mathrm{Int}_j\}$, for example $\{(-\infty,1.5],(1.5,2.5],\ldots,(4.5,\infty)\}$ in a five-point-scale case, predicted ratings are placed into the bins corresponding these intervals. By using these bins, $\Pr[r|v]$ is modeled by a multinomial distribution:

$$\Pr[\hat{r}|v] \approx \prod_{j=1}^{\#\mathrm{Int}} \left[\frac{\sum_{(x_i,y_i)\in\mathcal{D}^{(v)}} \mathbb{I}[\hat{r}(x_i,y_i,v)\in\mathrm{Int}_j]}{|\mathcal{D}^{(v)}|}\right]^{\mathbb{I}[r\in\mathrm{Int}_j]}, \tag{8}$$

where $\mathbb{I}[r \in \mathrm{Int}]$ is an indicator function and $\#\mathrm{Int}$ is the number of intervals. We refer to this model as mi-hist, which is an abbreviation of *mutual information modeled by a histogram model*.

However, because this model has discontinuous points, we develop a second new approach, which is to model $\Pr[\hat{r}|v]$ by a single normal distribution, which is continuous and easy to handle. Formally,

$$\Pr[\hat{r}|v] \approx \mathrm{Normal}\left(\hat{r};\, \mathbb{M}_{\mathcal{D}^{(v)}}(\{\hat{r}\}), \mathbb{V}_{\mathcal{D}^{(v)}}(\{\hat{r}\})\right), \tag{9}$$

where $\mathbb{V}_{\mathcal{D}}(\{\hat{r}\})$ is a sample variance over predicted ratings $\hat{r}_i$ from samples in $\mathcal{D}$. We refer to this model as mi-normal, which is an abbreviation of *mutual information modeled by a normal distribution model.*

Unfortunately, it is not easy to derive an analytical form of gradients for these neutrality terms. This is because the discretization is a discontinuous transformation in the mi-hist case, and $\Pr[\hat{r}]$ is a normal mixture, which is not a member of an exponential family, in a mi-normal. We therefore adopt the Powell optimization method for this class of neutrality terms, because it can be applied without computing gradients. However, this optimization method is too slow to apply to a large data set, and its lack of scalability is a serious deficit. In our implementation, these methods failed to complete the processing of 100k data in several days, whereas the methods described in the next section could process this dataset in minutes.

3.3.2 Calders-Verwer's Discrimination Score

To develop a neutrality term whose gradients can be derived in analytical form, we borrowed an idea in discrimination-aware data mining [13]. We here introduce Calders and Verwer's approach used in [2]. They proposed a score to measure the degree of socially discriminative decision, which is here referred by a *CV score*. This CV score is defined as the difference between distributions of target variable given $V = 0$ and $V = 1$.

$$\Pr[R|V = 0] - \Pr[R|V = 1]. \tag{10}$$

To reduce the influence of V on R, they tried to learn a classification model that would make the two distributions, $\Pr[R|V = 0]$ and $\Pr[R|V = 1]$, similar by causing the CV score to approach zero. It is easy to show that this process enforces the statistical independence between V and R [6]. Based on this idea, we design two types of neutrality terms the would make the two distributions $\Pr[R|V = 0]$ and $\Pr[R|V = 1]$ similar.

We design the first type of neutrality term so as to match the first-order moment of the two distributions, i.e., the means. It is formally defined as

$$-(\mathbb{M}_{\mathcal{D}^{(0)}}(\{\hat{r}\}) - \mathbb{M}_{\mathcal{D}^{(1)}}(\{\hat{r}\}))^2. \tag{11}$$

We refer to this neutrality term as m-match, which is an abbreviation of *mean matching*. The second type is designed to constrain so that the same ratings are predicted for the same value pair, x and y, irrelevant of the viewpoint values. This neutrality term is formally defined as

$$-\sum_{(x_i,y_i)\in\mathcal{D}} (\hat{r}(x_i, y_i, 0) - \hat{r}(x_i, y_i, 1))^2. \tag{12}$$

We refer to this neutrality term as r-match, which is an abbreviation of *rating matching*.

Because both types of neutrality terms are simple quadratic polynomials, it is very easy to derive analytical forms of their derivatives. We hence used a conjugate gradient method for these neutrality terms in optimization, which is much more efficient than the Powell method. Even if the size of data set becomes larger, more scalable optimizers, e.g., a stochastic gradient method, can be used because the gradients can be analytically calculated.

These terms have the additional merit of being less frequently trapped by local minima, because they are simple quadratic formulae. Conversely, it is not straightforward to extend these CV-score-based neutrality terms so that they are applicable to the case in which a viewpoint variable is multivariate discrete or continuous, as in mutual-information-based neutrality terms. When comparing m-match and r-match, the computation time for r-match is roughly twice that for m-match, because a rating prediction function must be evaluated for the cases of both $V = 0$ and $V = 1$ to compute r-match. r-match more strictly formulates neutrality than m-match. In the case of m-match, because the neutrality is enhanced on average over the user population, the neutrality of one user might be greatly enhanced, but that of the other might not. On the other hand, r-match is designed so that neutrality is uniformly enhanced almost everywhere over the domain of users and items. Unlike m-match, r-match treats counterfactual cases. For example, when the gender of a user is a viewpoint, even though the gender does not change, ratings in such a counterfactual case must be computed for using the r-match term. This fact may be semantically improper.

4. EXPERIMENTS

We implemented our information-neutral recommender system and applied it to a benchmark data set. We examined the four types of neutrality terms proposed in section 3.3.

4.1 Data Set

We used a Movielens 100k data set [4] in our experiments. Unfortunately, neither of the mutual-information-based methods in section 3.3.1, mi-hist and mi-normal, were able to process this entire data set. Therefore, we shrank the Movielens data set by extracting events whose user ID and item ID were less than or equal to 200 and 300, respectively. For scalable m-match and r-match methods, we applied a larger data set as described in section 4.4. This shrunken data set contained $9,409$ events, 200 users, and 300 items. The purpose of experiments on this small set was to compare the characteristics of all four neutrality terms. The mutual-information-based methods more strictly modeled the distribution over R and V than the CV-score-based methods described in section 3.3.2, m-match and r-match. If the CV-score-based methods behaved similarly to the mutual-information-based methods, we would be able to conclude that CV-score-based methods can enhance the neutrality and are scalable.

We tested the following two types of viewpoint variable. The first type of variable, Year, represents whether a movie's release year is newer than 1990, which is part of the item component of an event. In [10], Koren reported that older movies have a tendency to be rated more highly, perhaps because masterpieces are more likely to survive, and thus the set of older movies has more masterpieces. When adopting Year as a viewpoint variable, our recommender enhances the neutrality from this masterpiece bias. The second type of variable, Gender, represents the user's gender, which is part of the user component of an event. We expect that the movie ratings would depend on the user's gender.

4.2 Experimental Conditions

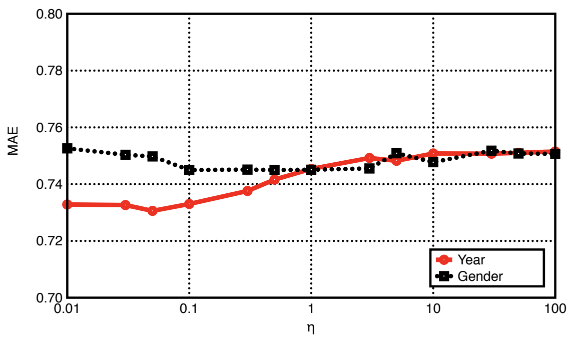
We optimized an objective function (5) with neutrality terms mi-hist or mi-normal by the Powell method, and that with terms m-match or r-match by the conjugate gradient method implemented in the SciPy package [16]. To initialize the model parameters, events in a training set, $\mathcal{D}$, were first divided into two sets according to their viewpoint values. For each value of a viewpoint variable, the parameters were initialized by minimizing an objective function of an original probabilistic matrix factorization model (equation (3)). For convenience in implementation, a loss term of an objective was re-scaled by dividing it by the number of training examples, and an L_2 regularizer was scaled by dividing it by the number of parameters. The four types of neutrality terms were re-scaled so that the magnitudes of these terms became roughly equal. Because the original rating values are $1, 2, \ldots, 5$, we adopted five bins $(-\infty, 1.5], (1.5, 2.5], \ldots, (4.5, \infty)$ for the mi-hist term. We use a regularization parameter $\lambda = 0.01$ and the number of latent factors, $K = 1$, which is the size of vectors $\mathbf{p}^{(v)}$ or $\mathbf{q}^{(v)}$. It should be notice that this data set was so small that the prediction performance was degraded if $K > 1$. Though in the case without cross term, i.e., $K = 0$, the performance was better than the case where $K = 1$, but we tested the model having the minimum cross terms. Our experimental codes are available at `http://www.kamishima.net/inrs/`.

We evaluated our experimental results in terms of prediction errors and the degree of neutrality. Prediction errors were measured by the mean absolute error (MAE) [5]. This index was defined as the mean of the absolute difference between the observed rating values and predicted rating values. A smaller value of this index indicates better prediction accuracy. To measure the degree of neutrality, we adopted mutual information between the predicted ratings and viewpoint values. The smaller mutual information indicates a higher level of neutrality. Mutual information is normalized into the range $[0, 1]$ by employing the geometrical mean as described in [6]. Note that the distribution $\Pr[\hat{r}|v]$ is required to compute this mutual information, and we used the same histogram model as in equation (8). We performed a five-fold cross-validation procedure to obtain evaluation indices of the prediction errors and the neutrality measures.

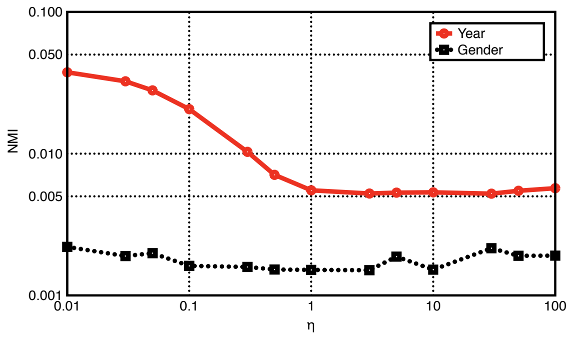
4.3 Experimental Results

Experimental results for the four types of neutrality terms are shown in Figure 1. The MAE was 0.903, when the rating being offered was held constant at 3.74, which is the mean rating over all ratings in the training data. This approximately simulates the case of randomly recommending items, and can be considered as the most unbiased and unintentional recommendation. We call this case *random prediction*. On the other hand, when applying the original probabilistic matrix factorization model (equation (2)), the MAE was 0.759. Because the trade-off for enhancing the neutrality generally worsens the prediction accuracy the accuracy as discussed in section 2.2, this error level can be considered as the lower bound. We call this case *basic prediction*.

In Figures 1(a) and (c), the prediction errors were better than random predictions. Overall, the increase of MAEs as the neutrality parameter, η, was not very great in any of the neutrality terms. The errors for the r-match term sometimes decreased even if η was increased. As described in section 4.2, the model without cross terms better performed. We think that the cross term effects would be eliminated by the strong restriction of the r-match terms, and MAEs were improved. Turning to Figure 1(b) and (d), the results obtained with the r-match term and with the other three terms were clearly contrasted. The three terms, mi-hist, mi-normal, and m-match, yielded successfully enhanced neutrality for the Year data, but less enhanced neutrality for the Gender data. Conversely, the r-match term was able to enhance neutrality for the Gender data, but it failed to do so for the Year data. We expected that this distinction was caused by the original independence between predicted ratings and viewpoint values. By comparing the NMIs at $\eta = 0.01$ of Figures 1(b) and (d), it was found that the dependence between ratings and viewpoint values for the Year data was larger that for the Gender data. Additionally, as described in section 3.3.2, while the r-match term is designed so that neutrality is uniformly enhanced over the domain of users and items, the other three terms are designed so as to enhance neutrality on average. In the case of the Year data, the three terms could enhance neutrality on average because the neutrality was low when η was small. However, the restriction of the r-match term was expected to be too strong for this data set. On the other hand, because the averaged neutrality for the Gender data was high at the beginning, the three terms failed to improve the neutrality, but the stronger neutrality-enhancement ability of the r-match would be effective in this case.



(a) MAE for the Year and Gender data sets



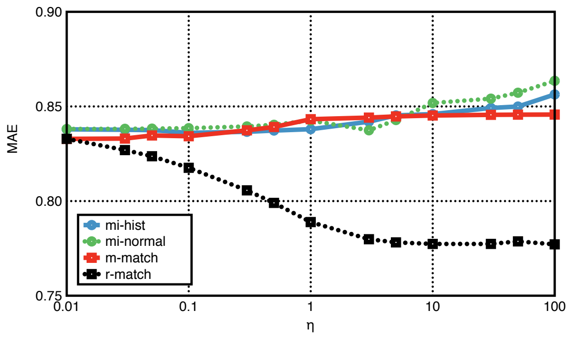
(b) NMI for the Year and Gender data sets

Figure 3: Changes of prediction errors and neutrality measures

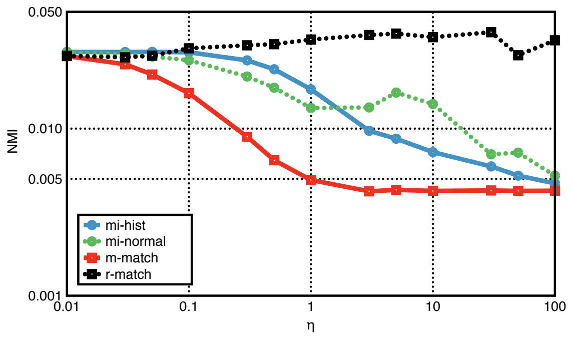
To further investigate this phenomenon, we show the changes of mean predicted ratings in Figure 2. Two types of neutrality terms, m-match and r-match, were examined. First, we focus on the case where $\eta = 0.01$, in which the neutrality term was less influenced. By comparing Figures 2(a) and (b), the difference between the mean ratings for old and new movies was much larger than the difference between the mean ratings rated by male and female users. In particular, while the former difference was 0.36, the latter difference was 0.024. This result again indicates that a higher level of neutrality is achieved for the Gender data than for the Year data. For the Year data, the m-match term successfully reduced the difference of two means as the increase of η, but the r-match term failed to do so. For the Gender data, both terms failed to reduce the difference between the two means, because the difference was already small and constraint terms were not effective.

4.4 Experiments on a Larger Data set

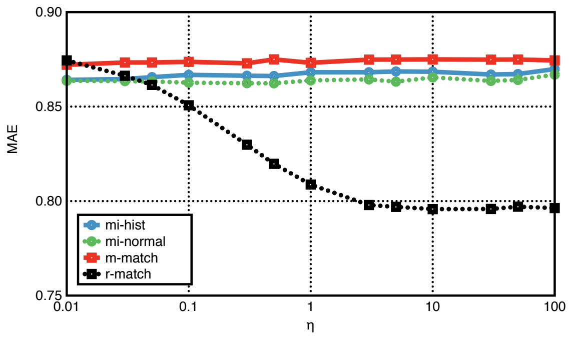
To show that our new neutrality terms are applicable to larger data sets, we made an INRS on the entire Movielens 100k data set, which contains 10 times as much data as the data set in the previous section. In our preliminary work [7], a data set of this size could not be processed. MAE of random and basic predictions for this data set were 0.945 and 0.750, respectively. We adopted the m-match neutrality term, and the other conditions were set as in section 4.2 except for $K = 3$. Figure 3 shows the changes of the MAE and NMI according to the increase of η. Trends similar to those in Figure 1 were observed. While neutrality was successfully enhanced without sacrificing prediction errors for the Year data, the m-match term was not effective for



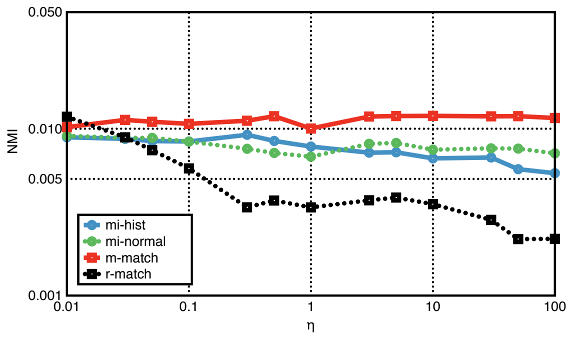
(a) Prediction error (MAE) for Year data



(b) Degree of neutrality (NMI) for Year data



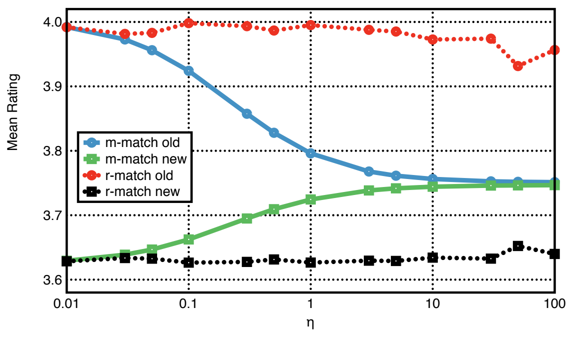
(c) Prediction error (MAE) for Gender data



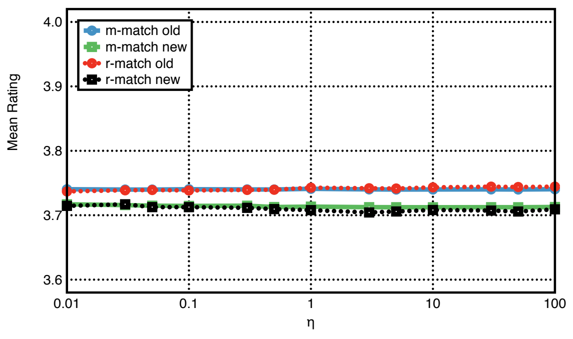
(d) Degree of neutrality (NMI) for Gender data

Figure 1: Changes of the degrees of neutrality accompanying the increase of a neutrality parameter

NOTE : Subfigures (a) and (b) are results on the Year data set, and Subfigures (c) and (d) are results on the Gender data set. Subfigures (a) and (c) show the changes of prediction errors measured by the mean absolute error (MAE in a linear scale). A smaller value of this index indicates better prediction accuracy. Subfigures (b) and (d) show the changes of the normalized mutual information (NMI in a log scale). A smaller NMI indicates a higher level of neutrality. The X-axes (log-scale) of these figures represent the values of a neutrality parameter, η, which balance the prediction accuracy and the neutrality. These parameters were changed from 0.01, at which the neutrality term was almost completely ignored, to 100, at which the neutrality was strongly enhanced.



(a) Year



(b) Gender

Figure 2: Changes of mean predicted ratings accompanying the increase of a neutrality parameter

NOTE : In both figures, the X-axes (log-scale) represent the values of a neutrality parameter, η, and the Y-axes represent mean predicted ratings for each case with a different viewpoint value. Subfigure (a) shows mean the predicted ratings when the viewpoint variable is Year. Means for the movies before 1990 were designated as "old," and those after 1991 were designated as "new." Subfigure (b) shows the mean predicted ratings when the viewpoint variable is Gender. Means of the ratings given by males and females were represented by "M" and "F," respectively.

the Gender data.

Finally, we should comment on the computational time. Generally, terms based on mutual information were much slower than those based on CV score. This is because analytical forms of gradients can be derived for the m-match and r-match. In comparing the two terms, m-match and r-match, the former is found to be faster, as described in section 3.3.2. Empirically, as η increased, the convergence of optimizers became slower, because the neutrality terms were not smooth compared to the loss term and harder to optimize. The influence of the increase of η was more serious for the r-match than for the m-match.

5. RELATED WORK

We adopted techniques for fairness-aware or discrimination-aware data mining to enhance the neutrality. Fairness-aware data mining is a general term for mining techniques designed so that sensitive information does not influence the mining results. Pedreschi et al. first advocated such mining techniques, which emphasized the unfairness in association rules whose consequents include serious determinations [13]. Another technique of fairness-aware data mining focuses on classification designed so that the influence of sensitive information on classification results is reduced [8, 2]. These techniques would be directly useful in the development of an information-neutral variant of content-based recommender systems, because content-based recommenders can be implemented by standard classifiers.

Because information-neutral recommenders can be used to avoid the exploitation of private information, these techniques are related to privacy-preserving data mining [1]. To protect private information contained in rating information, dummy ratings were added [18].

6. CONCLUSION

In this paper, we proposed an information-neutral recommender system that enhances neutrality from the viewpoint specified by a user. This system is useful for alleviating the filter bubble problem. We then developed an information-neutral recommendation algorithm by introducing several types of neutrality terms. Because the neutrality term in our preliminary work had poor scalability, we proposed a new and more efficient neutrality term. Finally, we demonstrated that neutrality in recommendation could be enhanced by our algorithm without sacrificing the prediction accuracy.

There are many functionalities required for this information-neutral recommender system. We plan to explore the other types of neutrality terms that can more exactly evaluate the independence between a target variable and a viewpoint variable while maintaining efficiency. Because viewpoint variables are currently restricted to binary type, we also try to develop a neutrality term that can deal with a viewpoint variable that is multivariate discrete or continuous. Though our current technique is mainly applicable to the task of predicting ratings, we will develop another algorithm for the task of recommending good items.

7. ACKNOWLEDGMENTS

We would like to thank for providing a data set for the Grouplens research lab. This work is supported by MEXT/JSPS KAKENHI Grant Number 16700157, 21500154, 23240043, 24500194, and 25540094.

8. REFERENCES

[1] C. C. Aggarwal and P. S. Yu, editors. *Privacy-Preserving Data Mining: Models and Algorithms*. Springer, 2008.

[2] T. Calders and S. Verwer. Three naive bayes approaches for discrimination-free classification. *Data Mining and Knowledge Discovery*, 21:277–292, 2010.

[3] S. Forden. Google said to face ultimatum from FTC in antitrust talks. Bloomberg, Nov. 13 2012. ⟨`http://bloom.bg/PPNEaS`⟩.

[4] Grouplens research lab, university of minnesota. ⟨`http://www.grouplens.org/`⟩.

[5] A. Gunawardana and G. Shani. A survey of accuracy evaluation metrics of recommendation tasks. *Journal of Machine Learning Research*, 10:2935–2962, 2009.

[6] T. Kamishima, S. Akaho, H. Asoh, and J. Sakuma. Considerations on fairness-aware data mining. In *Proc. of the IEEE Int'l Workshop on Discrimination and Privacy-Aware Data Mining*, pages 378–385, 2012.

[7] T. Kamishima, S. Akaho, H. Asoh, and J. Sakuma. Enhancement of the neutrality in recommendation. In *Proc. of the 2nd Workshop on Human Decision Making in Recommender Systems*, pages 8–14, 2012.

[8] T. Kamishima, S. Akaho, H. Asoh, and J. Sakuma. Fairness-aware classifier with prejudice remover regularizer. In *Proc. of the ECML PKDD 2012, Part II*, pages 35–50, 2012. [LNCS 7524].

[9] Y. Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proc. of the 14th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, pages 426–434, 2008.

[10] Y. Koren. Collaborative filtering with temporal dynamics. In *Proc. of the 15th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, pages 447–455, 2009.

[11] E. Pariser. The filter bubble. ⟨`http://www.thefilterbubble.com/`⟩.

[12] E. Pariser. *The Filter Bubble: What The Internet Is Hiding From You*. Viking, 2011.

[13] D. Pedreschi, S. Ruggieri, and F. Turini. Discrimination-aware data mining. In *Proc. of the 14th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, pages 560–568, 2008.

[14] P. Resnick, J. Konstan, and A. Jameson. Panel on the filter bubble. The 5th ACM Conf. on Recommender Systems, 2011. ⟨`http://acmrecsys.wordpress.com/2011/10/25/panel-on-the-filter-bubble/`⟩.

[15] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems 20*, pages 1257–1264, 2008.

[16] Scipy.org. ⟨`http://www.scipy.org/`⟩.

[17] S. Watanabe. *Knowing and Guessing – Quantitative Study of Inference and Information*. John Wiley & Sons, 1969.

[18] U. Weinsberg, S. Bhagat, S. Ioannidis, and N. Taft. Blurme: Inferring and obfuscating user gender based on ratings. In *Proc. of the 6th ACM Conf. on Recommender Systems*, pages 195–202, 2012.

[19] M. Zhang and N. Hurley. Avoiding monotony: Improving the diversity of recommendation lists. In *Proc. of the 2nd ACM Conf. on Recommender Systems*, pages 123–130, 2008.

[20] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *Proc. of the 14th Int'l Conf. on World Wide Web*, pages 22–32, 2005.

Towards User Profile-based Interfaces for Exploration of Large Collections of Items

Claudia Becerra
Universidad Nacional de Colombia
Bogotá - Colombia
www.unal.edu.co
cjbecerrac@unal.edu.co

Sergio Jimenez
Universidad Nacional de Colombia
Bogotá - Colombia
www.unal.edu.co
sgjimenezv@unal.edu.co

Alexander Gelbukh
Instituto Politécnico Nacional,
Centro de Investigación en
Computación, Mexico, D.F
http://nlp.cic.ipn.mx/
gelbukh@cic.ipn.mx

ABSTRACT

Collaborative tagging systems allow users to describe and organize items using labels in a free-shared vocabulary (tags), improving their browsing experience in large collections of items. At present, the most accurate collaborative filtering techniques build user profiles in latent factor spaces that are not interpretable by users. In this paper, we propose a general method to build linear-interpretable user profiles that can be used for user interaction in a recommender system, using the well-known *simple additive weighting model* (SAW) for multi-attribute decision making. In experiments, two kinds of user profiles where tested: one from free contributed tags and other from keywords automatically extracted from textual item descriptions. We compare them for their ability to predict ratings and their potential for user interaction. As a test bed, we used a subset of the database of the University of Minnesota's movie review system—Movielens, the social tags proposed by Vig et al. (2012) in their work "The Tag Genome", and movie synopses extracted from the Netflix's API. We found that, in "warm" scenarios, the proposed tag and keyword-based user profiles produce equal or better recommendations that those based on latent-factors obtained using matrix factorization. Particularly, the keyword-based approach obtained 5.63% of improvement. In cold-start conditions—movies without rating information, both approaches perform close to average. Moreover, a user profile visualization is proposed arising an accuracy vs. interpretability tradeoff between tag and keyword-based profiles. While keyword-based profiles produce more accurate recommendations, tag-based profiles seems to be more readable, meaningful and convenient for creating *profile-based user interfaces*.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrievall** Information Search and Retrieval]: Selection process; H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces–*Collaborative computing*; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces

General Terms

Algorithms, Experimentation.

Keywords

Recommender systems, collaborative filtering, collaborative tagging systems, social tagging, user interfaces

1. INTRODUCTION

An approach for improving the exploration of large collections of items such as books (librarything.com), films (netflix.com), pictures (flickr.com), research papers (citeulike.com) and web bookmarks (del.icio.us) is the leveraging of collaborative information from the users. This approach allows the knowledge of certain individuals on certain items in the collection propagates towards other users. In this way, a self-generated collaborative intelligence guides users in their exploration by recommendations tailored to their preferences and away from dislikes.

Currently, collaborative filtering approaches derive user profiles and produce recommendations based primarily on user feedback whether explicit (e.g. ratings, "likes", tagging, reviews) or implicit (e.g. web logs). As the time goes by, user profiles grow while their preferences evolve. Generally, users are allowed to update their explicitly given information with the aim of adjusting their profiles to get better recommendations. In this scenario, when a user wants to update his (her) profile, it depends—for instance—on a large number of ratings making of this a difficult and even overwhelming task. The users should make a significant number of targeted edits in their profiles to obtain the desired effect. The situation worsens in systems based on implicit feedback where user profiles are not interpretable nor accessible by users.

Most of the state-of-the-art methods for collaborative filtering build user profiles projected in latent factor spaces. These latent factors reduce considerably the dimensionality of the user profiles providing more accurate recommendations at the expense of interpretability. Unfortunately, users cannot make modifications on these low-dimensional and highly informative profiles. A first step to tackle this issue could be the design of interfaces based on interpretable user profiles. For instance Lops et al. [16] proposed a system where the user profiles are defined in a space indexed by keywords automatically extracted from textual item descriptions —*keyword-based user profiles.* However, in many cases the number of extracted keywords is similar or even larger than the number of items in the collection making it difficult the interaction of users with their profiles.

Alternatively, user profiles can also be built using tags [2]—*tag-based user profiles*. These tags come from collaboratively tagging systems [29], which allows users in large collections to label items using a shared free vocabulary. As a result of this social indexing process [10], the system gradually collects a social index, which enables users to classify, visualize and query items in a way that is both personalized and social. Unfortunately, social indexes suffer of misspellings, typographical errors and extremely particular tags, making of them a noisy resource for the construction of meaningful user profiles. Sen et al. (2009) [23] proposed an entropy-based measure and a cleaning procedure for detecting a community-valuable tag set from a noisy social index. They obtained a clean set of 1,128 tags from nearly 30,000 different tags collected by the MovieLens[1] system during the year 2009. Clearly, this tag set has a more convenient size for designing user interfaces for customizing user profiles based on social tags.

In this paper, we propose a method based on matrices for building linear user profiles based either on social tags or on automatically extracted keywords. From the users' point of view, these profiles behave as a linear *simple aggregative weighting model SAW* [28], that is one of the most comprehensive method for multi-attribute decision making [12]. So, the proposed method discovers the prior weights, or the users' affinity coefficients with tags or keywords, that minimize the rating prediction error. These produced profiles—*SAW user profiles*—can be used either to invite users to interact with their own profiles or to explain the recommendations given by the system.

To evaluate the performance of SAW user the profiles, they were compared against user profiles based on latent factors obtained using matrix factorization techniques [15], [4]. This comparison was made in the rating prediction task for the movie domain. We observed that the proposed methods outperformed or reached similar results in cross-validation and cold-start evaluation settings (respectively) in comparison with strong baselines. That is the main contribution of this work: a collaborative method to obtain *simple aggregative weighting user profiles* without compromising rating prediction accuracy.

In addition, a visualization of user profiles is provided with the aim of analyzing the potential of SAW user profiles for the construction of user interfaces for recommender systems. In that visualization the profile of a single user is shown as a list of tags, or keywords, ranked by preference. We argue that the hypothetical user interaction with the top and the bottom of that list would provide a mechanism for updating his user profile with little effort. Simultaneously, the profiles of the nearest users are also shown as a collaborative resource for suggesting updates.

2. RELATED WORK

There have been several works that let users directly interact with keyword-based user profiles or tag-based user profiles. For example, the work of Pazzani and Billsus (1997) [9] is the earliest system that let users directly interact with their keyword-based user profiles. In that work, users directly assess the conditional probability of liking or disliking a resource given that a particular word is found in the resource's textual description. These user-provided conditional probabilities are used as priors to train a Naïve Bayes classifier that, using users' ratings, estimates the probability of liking or disliking the resource using keywords as resource features. They found that these prior profiles increase the accuracy of the recommendations obtained by the Naïve Bayes classifier, mainly in cold-start scenarios [21] when users have not yet given enough ratings.

[1] http://www.movielens.org

Another example is the work of Diederich & Iofciu (2006) [6]. In their work, users directly interact with manually build tag-based user profiles as a way to query the system for obtaining recommendations. They used the digital library DBLP[2], where items (research papers) are labeled with tags manually specified by the authors. In a first stage, the system prepares a tag-based author profile aggregating the tags associated to the works of the author (see Table 1). Then, users can get recommendations of similar authors by using a query profile in which users change the coefficients assigned to the tags. With this query profile, the system recommends similar authors to the one queried using collaborative filtering approaches [11].

The main limitation of the above mentioned approaches is that only first order relations between user and resource are considered to build these profiles. Consequently these approaches are incapable to find new tags or keywords relevant to the profile. Other approaches integrate collaborative tagging information, and keywords found in textual descriptions of resources, in algorithms that outperform classic collaborative filtering approaches, but they sacrifice interpretability for accuracy [8, 9, 16]. Therefore in this work we propose a collaborative method to generate linear user profiles in interpretable spaces that can be inspected and eventually modified by users, without accuracy sacrifices.

Table 1: Example of a user profile in TBprofile[§]

User's personal library

Publication title	Tags (Keywords)
Magpie: supporting browsing and navigation on the semantic web	named entity recognition (NER), semantic web, …
Bootstrapping ontology alignment methods with APFEL	alignment, mapping, ontology, …
Swoogle: a search and metadata engine for the semantic web	rank, search, semantic web, …

Tag-based author profile

NER	Semantic web	SW services	alignment	Mapping	ontology	…
1	2	1	1	1	1	…

§ from Diederich & Iofciu (2006) [6]

3. METHODS

3.1 Matrix Factorization Overview

Probably, the most popular and accurate method used for product recommendation is matrix factorization [4], [15]. In this model the rating estimation $\hat{r}_{um}$ that a user u would give to an item m is estimated as an affinity measure between the user and the item, both characterized in a latent factor space with a pre-established dimensionality f. Formally:

$$\hat{r}_{um}=\vec{U}_{u\to\mathcal{R}^f}\cdot\left(\vec{M}_{m\to\mathcal{R}^f}\right)^T$$

Where $\vec{U}_{u\to\mathcal{R}^f}$ and $\vec{M}_{m\to\mathcal{R}^f}$ denotes the characterization of user u and item m in the latent factor space $\mathcal{R}^f$ respectively. Here, the used affinity measures is the dot product. If the components that characterize the user in the latent space $\mathcal{R}^f$ are denoted by

[2] http://www.informatik.uni-trier.de/~ley/db

$\vec{U}_{u\to\mathcal{R}^f} = [U_{u1}, U_{u2}, \ldots, U_{uf}]$, and the item vector components are denoted as $\vec{M}_{m\to\mathcal{R}^f} = [M_{m1}, M_{u2}, \ldots, M_{uf}]$, then the dot product can be rewritten as:

$$\hat{r}_{um}=\sum_{i=1}^{f}(U_{ui} \cdot M_{mi})$$

where the characterization of $\vec{U}_u$ and $\vec{M}_m$ vectors are found minimizing the prediction error e_{um}, which is calculated using the following expression:

$$e_{um} = \left(r_{um} - \sum_{i=1}^{f}(U_{ui} \cdot M_{mi})\right)^2$$

To avoid overfitting, it is common to introduce a regularization coefficient β that penalizes the norm of the user and item vectors. Thus, the regularized prediction error $\ddot{e}_{um}$ is defined as:

$$\ddot{e}_{um} = e_{um} + \beta\left(\left\|\vec{U}_{u\to\mathcal{R}^f}\right\|^2 + \left\|\vec{M}_{m\to\mathcal{R}^f}\right\|^2\right)$$

Finally, user and item vectors are found minimizing the regularized prediction error over the set of known ratings.

$$\min_{\vec{U}_u,\vec{M}_u} \sum_{r_{um}\in\mathbb{R}\,\wedge\, r_{um}\neq 0} \left(r_{um} - \sum_{i=1}^{f}(U_{ui} \cdot M_{mi})\right)^2 + \beta\left(\left\|\vec{U}_{u\to\mathcal{R}^f}\right\|^2 + \left\|\vec{M}_{m\to\mathcal{R}^f}\right\|^2\right)$$

In this expression, we organize the known ratings in the matrix $\mathbb{R}_{U\times M}$, of size $U \times M$, where U is the number of users and M is the number of items. In this matrix, unknown ratings r_{um} are assigned to 0, and known ratings are in the interval [1, 5].

3.2 Proposed Models

3.2.1 A Generic User Profiling Model

In spite of the fact that it could be considered incorrect[3], we will use the canonical form of matrix factorization to express the matrix of estimated ratings $\hat{\mathbb{R}}_{U\times M}$ as an affinity measure between the user profile matrix $\mathbb{U}_{U\times f}$ and the item profile matrix $\mathbb{M}_{M\times f}$, both characterized in the same latent factor space. Thus:

$$\hat{\mathbb{R}}_{U\times M} = \mathbb{U}_{U\times f} \cdot \left(\mathbb{M}_{M\times f}\right)^T$$

Now, we can generalize this affinity measure to any space of dimension X —denoted by $\mathcal{R}^X$— using the expression:

$$\hat{\mathbb{R}}_{U\times M} = \mathbb{U}_{U\times X} \cdot (\mathbb{M}_{M\times X})^T$$

Where $\mathbb{U}_{U\times X}$ is the $\mathcal{R}^X$*-based user profile* matrix and $\mathbb{M}_{M\times X}$ is the $\mathcal{R}^X$*-based item profile* matrix. The matrix of user profiles in the space $\mathcal{R}^X$, $\mathbb{U}_{U\times X}$, of size $U \times X$ can also be denoted as:

$$\mathbb{U}_{U\times X} = \begin{bmatrix} U_{11} & \cdots & U_{1X} \\ \vdots & \ddots & \vdots \\ U_{U1} & \cdots & U_{UX} \end{bmatrix} = \begin{bmatrix} \vec{U}_{1\to\mathcal{R}^X} \\ \vdots \\ \vec{U}_{U\to\mathcal{R}^X} \end{bmatrix}$$

Where U_{ux} represent the affinity coefficient between the user u and the x^{th} dimension in the space $\mathcal{R}^X$, for values of u in $\{1,..,U\}$ and values of x in $\{1,..,X\}$. In that notation, the vector $\vec{U}_{u\to\mathcal{R}^X}$ is the *X-based user profile* of user u in the space $\mathcal{R}^X$.

Similarly, the $\mathcal{R}^X$*-based user profile* matrix $\mathbb{M}_{M\times X}$ can be denoted as:

$$\mathbb{M}_{M\times X} = \begin{bmatrix} M_{11} & \cdots & M_{1X} \\ \vdots & \ddots & \vdots \\ M_{M1} & \cdots & M_{MX} \end{bmatrix} = \begin{bmatrix} \vec{M}_{1\to\mathcal{R}^X} \\ \vdots \\ \vec{M}_{M\to\mathcal{R}^X} \end{bmatrix}$$

Where M_{mx} denotes the relevance coefficient of the item m to the x^{th} dimension in the space $\mathcal{R}^X$, for values of m in $\{1,..,M\}$ and x in $\{1,..,X\}$. $\vec{M}_{m\to\mathcal{R}^X}$ represents the profile of the item m in the space $\mathcal{R}^X$.

Now, if we choose an interpretable space $\mathcal{R}^X$ in which the item profile matrix $\mathbb{M}_{M\times X}$ can be directly calculated, then all the user profiles in $\mathbb{U}_{U\times X}$ can be obtained by the following expression:

$$\mathbb{U}_{U\times X} = \mathbb{R}_{U\times M} \cdot ((\mathbb{M}_{M\times X})^T)^{-1}$$

Where $((\mathbb{M}_{M\times X})^T)^{-1}$ denotes the pseudo-inverse [18] of the transposed item profile matrix characterized in $\mathcal{R}^X$, and $\mathbb{R}_{U\times M}$ is the matrix of known ratings.

3.2.2 SAW User Profiles

Once the user profiles are obtained the estimated ratings $\hat{r}_{um}$ can be calculated with the expression:

$$\hat{r}_{um} = \sum_{x\in\{1,\ldots,|X|\}} U_{ux} \cdot M_{mx}$$

Therefore, from the point of view of decision making, it has the well-known canonical form of *the simple additive weighting method* (SAW) for multi-attribute decision making [13]. In this model, a linear discriminative function is used to appraise each resource assigning a value (weight) to each alternative. Alternatives with higher values are preferred over alternatives with lower values. Studies in the area [30], [1], [27] have shown that the intuitiveness of the SAW method makes it more preferable, for user direct interaction, than other less interpretable non-linear methods.

Thus, our proposed model, behave as a SAW model for decision making where: i) the appraisal of the resource is the rating of the resource $\hat{r}_{um}$; ii) ratings are expressed as a weighted linear combination of the resource features in the interpretable space $\mathcal{R}^X$; and iii) weights or the affinity coefficients U_{ux} are discovered by the proposed model.

In the following subsections 3.2.3 and 3.2.4, we will explain how this generic model can be applied in two different interpretable spaces, namely keywords and tags. Besides, we will also show how the proposed user profiles $\mathbb{U}$ can be used in combination with the matrix factorization model to obtain rating predictions (see

[3] It is important to keep in mind that, in order to calculate the approximation of $\mathbb{U}_{U\times f}$ and $\mathbb{M}_{M\times f}$ matrices, ratings $r_{um} = 0$ must be ignored in the expression to minimize. This is why in the recommendation study area, instead of using already implemented matrix decomposition methods, it is preferable to use optimization methods such us LBFGSB [5]. In these methods, the unknown ratings are expressly filtered from the training matrix $\mathbb{R}_{U\times M}$. Henceforth, the matrix notation will be used given the conceptual simplicity that it provides for the further discussion. However, all matrix factorizations will ignore unknown ratings r_{um}.

subsection 3.2.5). To clarify the notation used in the following sections, we will replace X for the specific size (dimensionality) of the space in which we will focus the discussion. Thus, $\mathcal{R}^W$ will be used instead of $\mathcal{R}^X$, to denote he space of keywords. Similarity, in subsection 3.2.4, the space defined by the tags will be denoted by $\mathcal{R}^T$.

3.2.3 Keyword-based User Profiles

As mentioned before, the proposed model that automatizes the process of construction of user profiles relies (in turn) in the construction of the item profiles. Therefore, the matrix $\mathbb{U}_{U\times W}$ (keyword-based user profiles) is calculated using the matrices $\mathbb{M}_{M\times W}$ (keyword-based item profiles) and $\mathbb{R}_{U\times M}$ (known ratings) using the following expression:

$$\mathbb{U}_{U\times W} = \mathbb{R}_{U\times M} \cdot ((\mathbb{M}_{M\times W})^T)^{-1}$$

Most of the content-based approaches that build keyword-based item profiles [16] use the vector space model [20] for representing the textual descriptions of the items as vectors $\vec{M}_{m\to\mathcal{R}^W}$. Components of this vector, denoted by M_{mw}, are values that quantify the relevance of the word w to the item m. Thus, a value close to 0 indicates that the word is not relevant to the item. Negative values can also be used if polarized relevance scores are available.

These relevance scores can be inferred from the occurrences of the words in the collection of textual descriptions of the items. The common practice to obtain relevance scores is to use the popular *tf-idf* term weighting scheme [14] or weights derived from the Okapi BM-25 retrieval formula [19]. These techniques prevent that common words get high relevance scores and promote less frequent words that occur systematically in particular textual descriptions.

3.2.4 Tag-based User Profiles

Analogously to the keyword-based profiles, the $\mathbb{U}_{U\times T}$ matrix with the *tag-based user profiles* is calculated in the same way:

$$\mathbb{U}_{U\times T} = \mathbb{R}_{U\times M} \cdot ((\mathbb{M}_{M\times T})^T)^{-1}$$

Where $\mathbb{M}_{M\times T}$ is the matrix with tag-based item profile vectors $\vec{M}_{M\to\mathcal{R}^T}$, in which the individual M_{mt} entries indicate the relevance of the tag t to the item m.

The tag-based item profiles $\vec{M}_{m\to\mathcal{R}^T}$ can be obtained using several techniques [16], [29]. The simplest approach consists in an item profile based on Boolean occurrences. That is, set $M_{mt} = 1$ when the tag t has been applied to the item m and $M_{mt} = 0$ otherwise. It is important to note that the proposed method to obtain the tag-based user profiles, using the pseudo-inverse, is equivalent to a linear regression. Therefore, the tags should be independent among them. That independence can be promoted grouping tags that are morphologically related using stemmers and lemmatizers. Lops et al. [17] went beyond grouping tags semantically related using WordNet synsets [7].

Item profiles with graded, instead of Boolean relevance scores can be obtained with more sophisticated methods. For instance, Vig et al. (2012) [26] obtained *the tag genome*—a tag-based item profile for movies—by training a support vector regressor [24]. The training data came from a survey applied to users from the MovieLens system. The users where asked to estimate the relevance of the tags applied on selected movies. With these answers and a set of features extracted from movie reviews, textual descriptions, metadata and tag applications, among others, they trained a regressor whose predictions were used as relevance scores.

3.2.5 Hybrid and Updatable Rating Estimation

The proposed method for generating the rating predictions is a combination of matrix factorization (subsection 3.1) and the user profiles proposed in subsections 3.2.3 and 3.2.4. The aim of the method is three fold. First, we look for rating predictions as good as the ones produced by matrix factorization. Second, the method should be hybrid, that is, a combination of the collaborative filtering approach of matrix factorization and the content information from keywords or tags. Third, the users should be able to edit their keyword-based (or tag-based) user profiles and the rating predictions must be updated with little computational cost. The method comprises four steps:

1. An initial matrix of rating estimations is obtained using matrix factorization: $\hat{\mathbb{R}}^0_{U\times M} = \mathbb{U}_{U\times f} \cdot \left(\mathbb{M}_{M\times f}\right)^T$.
2. An initial matrix of keyword-based user profiles is obtained: $\mathbb{U}^0_{U\times W} = \hat{\mathbb{R}}^0_{U\times M} \cdot ((\mathbb{M}_{M\times W})^T)^{-1}$.
3. The matrix $\mathbb{E}_{U\times W}$, containing users edition operations to their profiles (positive of negative differences) is added to obtain updated user profiles: $\mathbb{U}_{U\times W} = \mathbb{U}^0_{U\times W} + \mathbb{E}_{U\times W}$.
4. Estimations are obtain by: $\hat{\mathbb{R}}_{U\times M} = \mathbb{U}_{U\times W} \cdot (\mathbb{M}_{M\times W})^T$

These four steps can be expressed in a single expression:

$$\hat{\mathbb{R}}_{U\times M} = \left(\hat{\mathbb{R}}^0_{U\times M} \cdot ((\mathbb{M}_{M\times W})^T)^{-1} + \mathbb{E}_{U\times W}\right) \cdot (\mathbb{M}_{M\times W})^T$$

Note that $((\mathbb{M}_{M\times W})^T)^{-1} \cdot (\mathbb{M}_{M\times W})^T \cong \mathbb{I}_{M\times M}$ (the identity matrix) only when the item profiles are linearly independent among them. The contrary is the common case. Thus, this matrix multiplication infers the affinities among the items induced by the keywords content information. In a final post-processing step, the values on each row in the output matrix $\hat{\mathbb{R}}_{U\times M}$ are standardized in the interval $[-1,1]$. The final rating predictions are obtained adding to each estimated rating the average rating of the movie and the user's bias. The user bias is the average deviation of the user's ratings against the average of the entire set of ratings. The rating estimation using tag-based user profiles is the same but replacing $\mathbb{M}_{M\times W}$ by $\mathbb{M}_{M\times T}$.

4. EXPERIMENTATION

The experiments aim to evaluate the accuracy of the recommendations produced by the proposed methods. This section contains a comprehensive description of the data and the evaluation measure used to compare the proposed models against baselines.

4.1 Data

This subsection is intended to provide insight about how the used dataset was obtained and preprocessed. Besides we provide information about its content, size and distribution.

4.1.1 Movies Collaborative Data

The dataset of users, movies and ratings was obtained from a production database dump of the MovieLens system in April 2012. From this dataset, we extracted a subset filtering by the users and movies with more than 1,000 ratings. This filtering produced a subset of 200 users, 1,462 movies and 150,915 ratings. The rating scale in MovieLens is in the usual interval [1,5], having 5 as the maximum grade of preference. The distribution of ratings in our dataset is shown in Figure 1. The average number of ratings per movie is 101.6 (σ = 37.5), and per user is 742.5 (σ = 188.5).

4.1.2 *Textual Descriptions of the Movies*

Textual descriptions were obtained from the synopsis field in the movie records from the Netflix public API[4] during the year 2012. These texts were assigned to movies in the MovieLens dataset by a mapping obtained through a research collaboration with the GroupLens[5] research group.

These textual descriptions were represented in a vectorial bag-of-words model. The dimensionality of that representation was reduced with the aim of obtaining a vocabulary based on popularity and informativeness. Thus, a vocabulary of 5,848 words was obtained using the following series of preprocessing ad hoc actions: (1) all characters were converted to lowercase equivalents; (2) people first and last names were concatenated with the underscore character; (3) numeric tokens were removed; (4) 334 stop words taken from the source code of the *gensim*[6] framework were removed; (5) words occurring in less than 10 synopses and in more than the 95% of the synopses, were removed; and finally (6) all punctuation marks were cleaned.

The term weights used to register the relevance of a word in a synopsis vector were obtained with the Okapi BM25 retrieval formula [19] using the method proposed by Vanegas et al. [25]. Thus, the weight $w(p,d)$ of a word p in a document (synopsis) d is given by:

$$w(p,d) = log\left(\frac{M - df(p)}{M}\right)\frac{(k_1+1)tf(p,d)}{K + tf(p,d)}$$

$$K = k_1\left((1-b) + b\frac{dl(d)}{avdl}\right)$$

Where, $df(p)$ is the number of documents where p occurs, $M = 1{,}462$ is the number of movies, $tf(p,d)$ the number of occurrences of word p in the document d, and $avdl = 33$ is the average document length. The additional used parameters were $k_1 = 1.2$ and $b = 0.75$ (see [e]). A pair of examples of the resulting keyword vectors using the proposed method is shown in Table 2. The aggregation of vectors obtained from synopses produce the items profile matrix $\mathbb{M}_{M\times W}$, whose dimensions are $M = 1{,}462$ movies (rows) by $W = 5{,}848$ words (columns). This matrix is sparse, having only 0.518% of non-zero entries.

4.1.3 *Social Tags*

The tag set used to characterize the movies is the selection of tags proposed by Vig et al. in "The Tag Genome" [26]. This tag set is a subset of 1,128 tags out of nearly 30,000 unique tags freely applied by 416 users in the MovieLens system. This subset was obtained by removing tags with less than 10 applications, misspellings, people names and near duplicates. Thereafter, they selected the top 5% ranked tags with and entropy-based quality measure proposed by Sen et al. [22]. Only 1,081 tags from the tag genome's set occurred in the 1,462 movies in the item-profile matrix $\mathbb{M}_{M\times T}$.

There are 13,332 tag associations to the movies considered in this study. 1,370 movies have at least one tag associated with an average of 9.7 tags per movie (σ = 8.5). Besides, all tags were assigned at least to one movie. The distribution of the tag applications is considerably more uniform than the Zipf distribution. Thus, the 108 more frequent tags (10%) represent only the 42% of the tag associations. This can be roughly seen in Table 3, which shows tag samples selected from uniformly separated rank ranges. The association of movies and tags produce the items profile matrix $\mathbb{M}_{M\times T}$ (1,462 movies by 1,082 tags) with binary entries and a density of 0.844% (also very sparse).

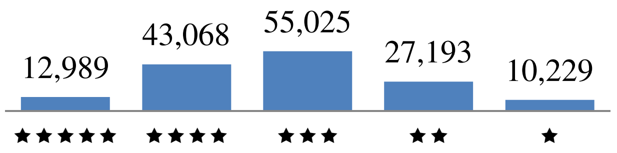


Figure 1: Rating distribution in the used subset of MovieLens

Table 2: Examples of keywords in Netflix's processed movie descriptions

Movie: "Bewitched (2005)" *will_ferrell* (0.237), *jack* (0.147), *update* (0.142), *samantha* (0.131), *sitcom* (0.131), *witch* (0.119), *nicole_kidman* (0.119), *convinced* (0.116), *michael_caine* (0.114), *right* (0.107), *hoping* (0.105), *know* (0.103), *career* (0.099), *perfect* (0.098), *doesnt* (0.097), *actor* (0.092), *make* (0.068), *film* (0.045)
Movie: "Rocky V (1990)" *burt_young* (0.249), *talia_shire* (0.242), *broke* (0.15), *upandcoming* (0.15), *shots* (0.15), *boxer* (0.15), *crooked* (0.142), *trainer* (0.136), *glory* (0.136), *accountant* (0.131), *ended* (0.131), *lifetime* (0.128), *memory* (0.124), *training* (0.124), *rocky* (0.121), *inspired* (0.107), *taking* (0.101), *career* (0.099), *left* (0.092), *series* (0.071), *takes* (0.063), *finds* (0.058)

Table 3: Samples of tags in the MovieLens tag set§

Rank	Sample tags
1-3	*based on a book* (194), *comedy* (182), *classic* (143)
9-12	*boring* (107), *70mm* (193), *romance* (98), *quirky* (91)
17-19	*sci fi* (78), *stylized* (64), *adventure*(62), *humorous*(62)
25-26	*crime* (53), *sequel*, *tense*, *violence*, *remake* (52)
34-35	*animation* (42), *politics*, *satirical*, *war*, *hilarious* (41)
42	*bittersweet* (34), *gay*, *historical*, *musical*, *suspense*
50	*forceful* (26), *military*, *satire*, *small town*, *very good*
59	*cult classic* (17), *dark humor*, *earnest*, *epic*, *japan* {17}
67	*action packed*(9), *alien*, *aviation*, *based on comic* {41}
75	*3d*(1), *adoption*, *airplane*, *alcatraz*, *arms dealer*: {80}

§ In parenthesis the number of movie associations to the tag; if missing, then it is the same as the precedent. The number of tags in the same rank is showed in curly brackets; if missing the listed tags are all the tags in that rank.

4.2 Experimental Setup

[4] http://developer.netflix.com
[5] http://www.grouplens.org
[6] http://radimrehurek.com/gensim

To evaluate the performance of the proposed methods we provided two scenarios of validation in 10 folds: cross validation and *product-cold-start* [24]. In the cross validation scenario, the ratings were divided in ten randomized folds. In each fold 90% of ratings were used for training and the remaining 10% was used for testing. In the product-cold-start scenario, the procedure for extracting the training and test datasets is the same, but all the ratings from the movies in the test set are removed.

The evaluation measure to assess the accuracy of the recommendations is root-mean-square error (RMSE) defined as:

$$RMSE = \sqrt[2]{\frac{\sum_{\{r_{um}\}\in test}(\hat{r}_{um} - r_{um})^2}{|test|}}$$

Where $test$ is the test set of the ratings and $|test|$ its cardinality. Given that the methods proposed in section 3 provide rating estimations standardized in $[-1,1]$ interval, $\hat{r}_{um}$ is obtained adding to these estimation the average of all the training ratings and the user's bias. Similarly, the baseline for the cold-start test scenario is a simple recommender system that predicts ratings based only on the average of all the training ratings plus the user's bias. The baseline method for the "warm"-start scenario is the recommender system based on matrix factorization presented in subsection 3.1. In all experiments, the number of latent factors was set to 30, $\beta = 0.07$ and the objective function was minimized using the LBFGSB optimization method [5].

Note that the matrix factorization method cannot be applied in the cold-start scenario because movies without ratings cannot be represented in the latent factors space. Consequently, for this scenario, the method proposed in subsection 3.2.5 uses $\mathbb{R}_{U\times M}$ instead of $\widehat{\mathbb{R}}^0_{U\times M}$ in the second step and the first step must be skipped.

5. RESULTS AND DISCUSSION

5.1 Recommendations Accuracy

The results of our experiments are presented in Table 4. The first two rows show the results for the proposed baseline methods for each one of our test settings. The remaining two rows show the results obtained by the proposed methods presented in subsection 3.2.4. For each system, the "RMSE" columns present the average for the 10 folds and the columns labeled with "σ" reports the standard deviation.

Table 4: Rating Prediction Results

METHOD	COLD START		WARM START	
	RMSE	σ	RMSE	σ
System average+user's bias	1.065	0.022	-	-
Matrix factorization	-	-	0.995	0.010
Keyword-based user prof.	1.052	0.015	0.939	0.016
Tag-based user profiles	1.062	0.021	0.985	0.012

Regarding the "warm" scenario (i.e. cross validation), the obtained results show that the two proposed methods based on user profiles outperformed the baseline matrix factorization method. Particularly, the margin obtained by the keyword-based user profile system was clearly significant, being more than 3 standard deviations apart. Clearly, the proposed methods reached a performance level in the state of the art for the rating prediction task. Unlike matrix factorization, our recommendations were produced by a fully interpretable model suitable for better user interaction and better explanations.

The cold-start evaluation setting was clearly more challenging. Our systems barely overcame the proposed average-based baseline. However, the proposed tag and keyword-based systems have the potential to provide to the user mechanisms to get the system "warmer" with little effort. Accurate methods such as matrix factorization require a considerable number of initial ratings before starting to produce good predictions. In contrast, our methods provide a completely customizable user profile with just a small number of initial ratings.

Comparing the tag-based and keyword-based models, the results show that keyword-based user profiling performs better in "warm" conditions and slightly better in "cold" conditions

5.2 Visualizing User Profiles

In order to visualize the profiles, we selected the *User 156* from the fold 1 in our dataset. We must say that users in our data are completely anonymous. This user was manually chosen based on the user-to-user pairwise Pearson correlation matrix obtained from the keyword-based user profiles $\mathbb{U}_{U\times W}$. Comparing these correlations we observed that the *User 156* had high negative and positive correlations against the other users. So, we considered that the preferences and dislikes of this user was being shared by several users and rejected by others. Consequently, we considered him as an interesting candidate to be visualized. In Figure 2, the keyword-based user profile of the *User 156* is showed jointly with his 10-nearest users according to the user-to-user correlation matrix. The ranked list of keywords that this user prefers the most is shown on the left side. The right side shows the list of his most disliked keywords. The user profile is represented by the thick black line. In its turn Figure 3, shows the same plots but using tag-based user profiles instead of keywords.

Now it is possible to qualitatively compare a user keyword-based versus a tag-based profile. From this comparison we observe that *User 156*'s tag-based profile is more cohesive in comparison with the word-based profile. This cohesiveness can be observed by the semantic relatedness of the tag set. In this profile, 20 out of 40 tags preferred by *User 156* are related to action and teens movies. These tags are: *Dark hero*, *Effects*, *Explosions*, *Indiana jones*, *German*, *Drug addiction*, *Arms dealer*, *Weapons*, *Life & death*, *Videogame*, *First contact*, *Comic book adapt*, *Bond, 007 series*, *Stop motion*, *Fantasy world*, *Dreamworks*, *Video games*, *Harry potter*, *Emma Watson*. Regarding the keyword-based profile, the keyword set doesn't exhibit a clear pattern. Although we know that these particular observations cannot be generalized, we think that this observation opens an interesting research direction about the necessity of measuring the semantic cohesiveness of the produced profiles.

Concerning the potential of interaction we have not yet conducted any experiments with users, but it seems reasonable that users will understand the general interaction idea. It is expected that the users will be prone to experiment modifying their own profiles varying the level of preference or dislike for the more relevant tags or keywords in their profiles. Also, it seems that the feature of seeing the profile of similar users could motivate the desire to interact with the interface. That is because, showing other people behaviors and allows a kind of warm start with the system.

New concerns arise from the observations of these profiles. For instance, what should be done with "negative" tags that appear in the list of preferred tags of users? This situation is illustrated by the tag "boring!" in the *User 156*'s "likes" list.

Probably, this tag can be reasonable and predictive for some users, so, maybe it shouldn't be removed from the tag set. But trenchant criticisms of user tastes should be prevented. A possible alternative to this problem would be the use of a linear regression algorithm, similar to the one used in a previous work [3], that could estimate a weight for each tag for knowing if the tag is intrinsically positive or negative. Thus, if a tag has a negative connotation we could filter it from the list of "liked" tags.

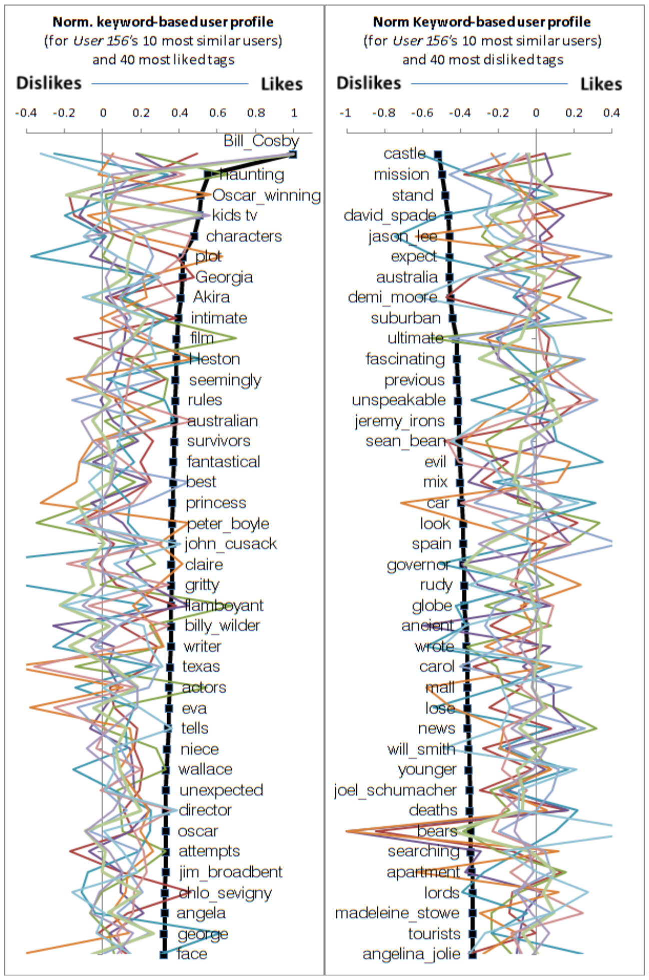


Figure 2: Keyword-based profile for User 156

6. CONCLUSIONS

We proposed a generic method to extract user profiles, in interpretable spaces, in which it is possibly to directly characterize items from the collection. The proposed user-profiling methods were indexed in two different spaces: keywords and tags. Besides the proposed models are suitable for user interaction in the user profile component.

The proposed user-profiling methods were evaluated in a subset of the MovieLens dataset and compared against strong baselines. It was concluded that in "warm" scenarios both methods produce recommendations with the same accuracy than those produced by matrix factorization methods. In a cold-start scenario, both methods performed slightly better than a recommender system based on average ratings.

In the warm-start scenario, when the keyword-based profiling and the tag-based profiling methods are compared, it was observed that keyword-based method was considerably more accurate than the matrix factorization method. The RMSE decremented by a 5.63% (more than 3 times σ), while the difference in the error with the tab-based method was only 1.00%. Consequently, it is possible to say that the proposed keyword-based method is able to improve the matrix factorization approach.

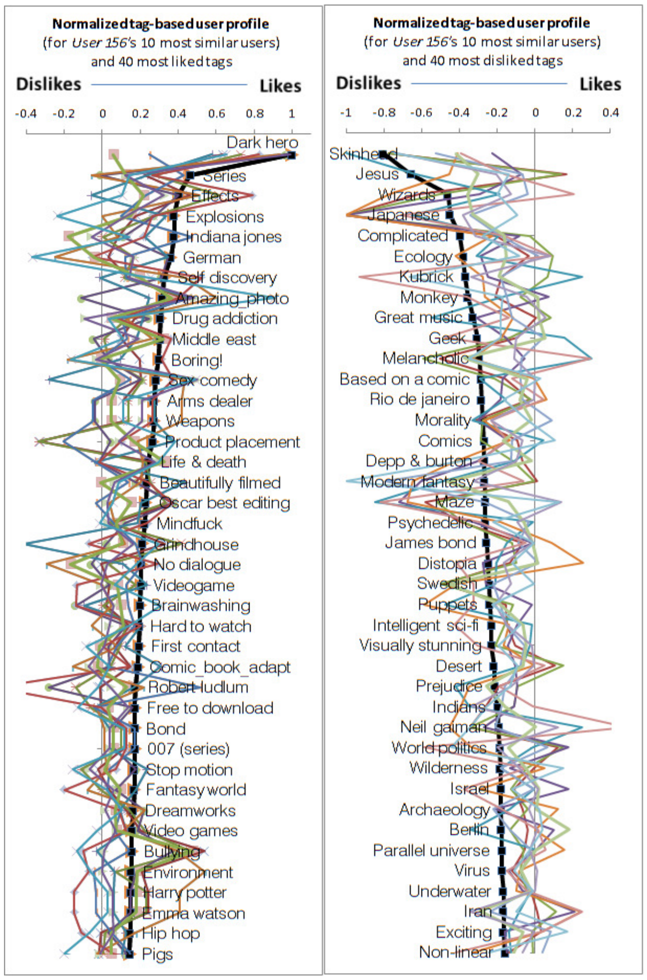


Figure 3: Tag-based profile for User 156

Regarding the proposed visualization of the keyword-based and the tag-based user profiles, we could observe that cohesion of the profile is an important measure to have into account when two profiles methods are compared. Non-cohesive profiles might be misunderstood by users leading them to avoid the interaction with those profiles. An interesting research question could be how to discriminate cohesive profiles, from non-cohesive profiles.

The proposed approach also contributed to a better classification of the content-based recommendation techniques, separating the user-profiling task from the item-profiling task, suggesting a uniform framework to share and compare the contributions made on each one of the tasks.

7. ACKNOWLEDGMENTS

Our especial thanks to Prof. John Riedl and Daniel Kluver from GroupLens, the University of Minnesota; Prof. Shilad Sen of the Macalester College; and Prof. Fabio Gonzalez of the Universidad Nacional de Colombia. The work was partially funded by the Colombian Department for Science, Technology and Innovation (Colciencias) via the grant 1101-521-28465 from "El Patrimonio Autónomo Fondo Nacional de Financiamiento para la Ciencia, la Tecnología y la Innovación, Francisco José de Caldas" and by the Universidad Nacional de Colombia via the grant DIB QUIPU: 201010016956. The third author recognizes the support from Mexican Government (SNI, COFAA-IPN, SIP 20131702, CONACYT 50206-H) and CONACYT–DST India (grant 122030 "Answer Validation through Textual Entailment").

7. REFERENCES

[1] Adomavicius, G., Manouselis, N. and Kwon, Y. 2011. Multi-Criteria Recommender Systems. *Recommender Systems Handbook*. 769–803.

[2] Man Au Yeung, C., Gibbins, N. and Shadbolt, N. 2008. A Study of User Profile Generation from Folksonomies. *SWKM* (2008).

[3] Becerra, C., Gonzalez, F. and Gelbukh, A. 2011. Visualizable and Explicable Recommendations Obtained from Price Estimation Functions. *Proceedings of the Human Decision Making in Recommender Systems* (2011), 27–34.

[4] Bell R.M., Koren Y. and C, V. 2007. The BellKor solution to the Net Flix Prize. *Technical report, AT&T Labs Research*. (2007).

[5] Byrd, R.H., Lu, P., Nocedal, J. and Zhu, C. 1995. A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput.* 16, 5 (Sep. 1995), 1190–1208.

[6] Diederich, J. and Iofciu, T. 2006. Finding Communities of Practice from User Profiles Based On Folksonomies. *Proceedings of the 1st International Workshop on Building Technology Enhanced Learning solutions for Communities of Practice* (2006).

[7] Fellbaum, C. ed. 1998. *WordNet An Electronic Lexical Database*. The MIT Press.

[8] De Gemmis, M., Lops, P., Semeraro, G. and Basile, P. 2008. Integrating tags in a semantic content-based recommender. *Proceedings of the 2008 ACM conference on Recommender systems* (New York, NY, USA, 2008), 163–170.

[9] Guan, Z., Wang, C., Bu, J., Chen, C., Yang, K., Cai, D. and He, X. 2010. Document recommendation in social tagging services. *Proceedings of the 19th international conference on World wide web* (New York, NY, USA, 2010), 391–400.

[10] Hassan-Montero, Y. and Herrero-Solana, V. 2006. Improving tag-clouds as visual information retrieval interfaces. *International Conference on Multidisciplinary Information Sciences and Technologies* (2006), 25–28.

[11] Herlocker, J.L., Konstan, J.A. and Riedl, J. 2000. Explaining collaborative filtering recommendations. (2000), 241–250.

[12] Hwang, C.L. and Yoon, K.M. 1981. Multiple Attribute Decision Making. Methods and Applications. *Springer-Verlag, NY.* (1981).

[13] Hwang, C.L. and Yoon, K.M. 1981. Multiple Attribute Decision Making. Methods and Applications. *Springer-Verlag, NY.* (1981).

[14] Jones, K.S. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*. 28, (1972), 11–21.

[15] Koren, Y., Bell, R. and Volinsky, C. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer*. 42, 8 (Aug. 2009), 30–37.

[16] Lops, P., Gemmis, M. and Semeraro, G. 2011. Content-based Recommender Systems: State of the Art and Trends. *Recommender Systems Handbook*. F. Ricci, L. Rokach, B. Shapira, and P.B. Kantor, eds. Springer US. 73–105.

[17] Lops, P., Gemmis, M., Semeraro, G., Musto, C., Narducci, F. and Bux, M. 2009. A Semantic Content-Based Recommender System Integrating Folksonomies for Personalized Access. *Web Personalization in Intelligent Environments*. G. Castellano, L. Jain, and A. Fanelli, eds. Springer Berlin Heidelberg. 27–47.

[18] Penrose, R. and Todd, J.A. On best approximate solutions of linear matrix equations. *Mathematical Proceedings of the Cambridge Philosophical Society*. null, 01, 17–19.

[19] Robertson, S. 2005. How Okapi Came to TREC. *TREC: Experiment in Information Retrieval*. MIT Press. 287–300.

[20] Salton, G., Wong, A.K.C. and Yang, C.-S. 1975. A vector space model for automatic indexing. *Commun. ACM*. 18(11), (1975), 613–620.

[21] Schein, A., Pennock, D. and Ungar 2002. Methods and metrics for cold-start recommendations. *SIGIR* (2002).

[22] Sen, S., Harper, F.M., LaPitz, A. and Riedl, J. 2007. The quest for quality tags. *Proceedings of the 2007 International ACM Conference on Supporting Group Work* (2007), 361–370.

[23] Sen, S., Vig, J. and Riedl, J. 2009. Learning to recognize valuable tags. *Proceedings of the 13th International Conference on Intelligent User Interfaces* (Sanibel Island, Florida, USA, 2009), 87–96.

[24] Smola, A.J. and Schölkopf, B. 1998. A Tutorial on Support Vector Regression,. *Royal Holloway College, London, U.K., NeuroCOLT Tech. Rep..TR 1998-030, 1998.* (1998).

[25] Vanegas, J.A., Caicedo, J.C., Camargo, J.E. and Ramos-Pollán, R. 2012. Bioingenium at ImageCLEF 2012: Textual and Visual Indexing for Medical Images. *CLEF (Online Working Notes/Labs/Workshop)* (Rome, Italy, 2012).

[26] Vig, Jesse, Sen, S. and Riedl, J. The Tag Genome: Encoding Community Knowledge to Support Novel Interaction. *ACM Transactions on Interactive Inteligent Systems*. 2, 3.

[27] Yeh, C.H. 2002. A problem based selection of multi-attribute decision-making methods. *International Transactions in Operational Research*. 9, 2 (Mar. 2002), 169–181.

[28] Yoon, K. and Hwang, C. 1995. Multiple Attribute Decision Making. An introduction. *Sage university papers series, no. 07-104. Thousand Oaks, CA: Sage Publications.* (1995).

[29] Zhang, Z.-K., Zhou, T. and Zhang, Y.-C. 2011. Tag-Aware Recommender Systems: A State-of-the-Art Survey. *Journal of Computer Science and Technology*. 26, 5 (Sep. 2011), 767–777.

[30] Zopounidis, C. and Doumpos, M. 2002. Multicriteria classification and sorting methods: A literature review. *European Journal of Operational Research*. 138, 2 (Apr. 2002), 229–246.

Selecting Gestural User Interaction Patterns for Recommender Applications on Smartphones

Wolfgang Wörndl
TU München
Boltzmannstr. 3
85748 Garching
Germany
woerndl@in.tum.de

Jan Weicker
TU München
Boltzmannstr. 3
85748 Garching
Germany
weicker@in.tum.de

Béatrice Lamche
TU München
Boltzmannstr. 3
85748 Garching
Germany
lamche@in.tum.de

ABSTRACT

Modern smartphones allow for gestural touchscreen and free-form user interaction such as swiping across the touchscreen or shaking the device. However, user acceptance of motion gestures in recommender systems have not been studied much. In this work, we investigated the usage of gestural interaction patterns for mobile recommender systems. We designed a prototype that implemented at least two input methods for each available function: standard on-screen buttons or menu options, and also a gestural interaction pattern. In a user study, we then compared what input method users would choose for a given function. Results showed that gesture usage depended on the specific task. In general, users preferred simpler gestures and rarely switched their input method for a function during the test.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – *Input devices and strategies, Interaction styles*

General Terms

Design, Experimentation, Human Factors.

Keywords

user interfaces, mobile applications, recommender systems, user study, gestural interaction.

1. INTRODUCTION

Recommender systems recommend movies, restaurants or other items to an active user based on ratings of items or other information about users and items. Recently, the focus in recommender systems research has been changing from investigating algorithms to studying the user experience [1]. This is especially true in mobile scenarios, for example on smartphones. Mobile information access suffers from limited resources regarding input capabilities, displays and other restrictions of small mobile devices. Therefore, user interfaces for mobile recommender systems have to be adapted to the specific properties of mobile devices [2].

The aim of this project is to study gestural interaction patterns for mobile recommender systems on smartphones, such as swiping across the touchscreen, or shaking the device. The specific goal of the work described in this paper is to map recommender functions - such as initiating a search for recommended items or rating an item - to reasonable gesture and motion interaction patterns. We designed a prototype to allow comparing user interface options and conducted a user study to find out which interaction patterns users would select when given a choice.

October 12–16, 2013, Hong Kong, China.
Paper presented at the 2013 Decisions@RecSys workshop in conjunction with the 7th ACM conference on Recommender Systems.

2. BACKGROUND

2.1 Gestural User Interaction Patterns

Saffer [3] distinguishes between two different forms of gestural interaction: touchscreen and free-form. Touchscreen gestures allow users to tap on the screen, either using on-screen buttons or other interface elements, e.g. sliders. Free-form gestures do not require the user to actively touch the screen but to move the devices to initiate functions. Current mobile devices offer several sensors that enable motion detection such as accelerometers and gyroscopes. The following touchscreen and free-form gestures are commonly used in mobile applications (Fig. 1).

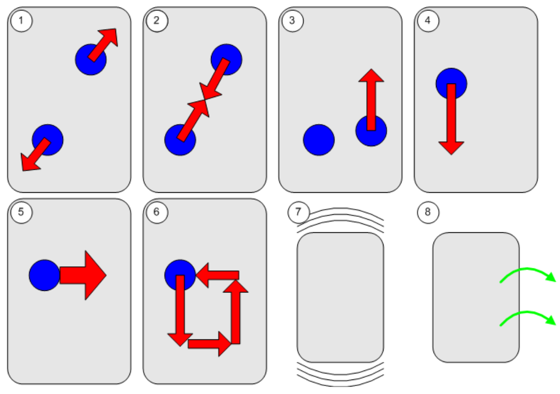


Figure 1. A visualization of how the different gestures are performed. Circles represent touches by fingers, arrows indicate movement. (1) Spread, (2) Pinch, (3) One-Finger-Hold Pinch, (4) Fling, (5) Flick/Swipe, (6) Rectangular Pattern, (7) Shake Device, (8) Tilt Device.

Single Tap is a brief one-finger tap on the screen and used in virtually every application to interact with on-screen buttons and similar interface objects. *Double Tap* means to tap the screen twice in rapid succession with one finger. *Pinch/Spread* is a two-finger gesture. The user places two fingers on the screen and moves them together (Pinch) or away from each other (Spread). This is most commonly used for zooming in (Spread) and out (Pinch). *One-Finger-Hold Pinch* is a more complex two-finger gesture. In this case, one finger rests on the screen, while a second finger moves on the screen to adjust a slider or other numerical value, for example.

Slide means to move a single finger over the screen in a continuous motion. *Slide* is generally used for dragging objects like sliders and slowly scrolling through views exceeding the screen's dimensions. *Fling* is a quick, long movement of one finger in one direction and can also be used for quickly scrolling through list views. *Flick* (or *Swipe*) is a shorter gesture similar to the longer *Fling* and commonly used as *Swipe-To-Delete* in file systems: a *Flick* gesture performed on an item generally deletes this item from a list. Another usage is moving to the next screen, resembling turning pages in a book. *Shake Device* and *Tilt Device* (along x, y or z axis) are free-form motion gestures with no screen interaction required.

Technically, any touch pattern can be drawn on the screen using one or more fingers, e.g. a rectangular pattern. However, finding the balance between gesture detection precise enough to distinguish different patterns, and vague enough to allow for user errors when drawing the patterns is difficult. In addition, explaining complex patterns to the user is challenging and therefore, complex patterns are rarely used in mobile applications.

2.2 Related Work

Previous research on the usage of gestures in mobile scenarios focused on the user acceptance of motion gestures in general and hardly applied these techniques for the interaction with recommender systems. In own previous work, we designed a minimalistic user interface for a map-based recommender based on gestural interaction, but for the larger screens of tablets [4].

Cho et al. propose a photo browsing system for mobile devices. They compared three types of interaction: a tilt-based interaction technique, an iPod wheel and a button-based browser to browse and search photos efficiently. The results show that the tilting technique is comparable to the controllability of buttons, more interesting than the other techniques and performed better than the iPod wheel [5]. Negulescu et al. examined the cognitive demands of motion gestures, taps and surface gestures. They show that these three techniques do not differ in reaction time. Moreover they found out that motion gestures result in much less time spent looking at the smartphone during walking than does tapping on the screen. Therefore motion gestures are advantageous in certain scenarios [6]. Rico and Brewster applied a different focus on motion gestures for mobile devices. They found out that location and audience have a significant influence on a user's willingness to interact with a mobile device by using motion gestures [7].

3. DESIGNING THE TEST APPLICATION

3.1 Overview

We implemented the prototype application for Android 2.2 (Froyo) and tested it on a Google Nexus One smartphone with Android. The goal of the test application was to provide different input methods for functions typically found in recommender systems to test which interaction patterns the user would chose in the successive study. The selection of functions in our application is not really specific to mobile recommenders and considers recommenders in a wider sense, i.e. taking also "search" applications into account. The scenario for the prototype is a movie search and recommendation application that resembles the *Internet Movie Database (IMDb)* mobile application (see http://www.imdb.com/apps).

We provided at least two different input methods for each application function, either

- on-screen buttons,
- menu options (the user has to select a specific "menu" option[1] to show additional buttons), or
- gestural interface options (cf. Section 2.1).

The next subsection describes considerations for mapping gestures to application-specific functions.

3.2 Considerations for Mapping Gestures to Application-Specific Functions

Single Tap is commonly used for interaction with on-screen interface objects and should not be used for other application-specific purposes. The same applies to *Slide* and *Fling* for scrolling screens or dragging objects. Contrariwise, *Double Tap* is not bound to any standard features and thus application-specific features can be mapped to it. As *Pinch/Spread* is generally used for zooming, mapping it to other application features may be confusing as well. However, the *One-Finger-Hold Pinch (OFHP)* variation of this gesture is applied in our application.

Since no screen interaction is necessary for the free-form gesture *Shake Device*, this gesture may be used independently from any interface restrictions, for example for application-wide functions. An application-wide function can be called at any time, regardless of the current screen of the application, e.g. the "home" button on most mobile systems. Functions depending on viewing items on-screen may not be viable for use with *Shake Device*, since shaking the screen makes focusing on displayed objects on the screen harder. The nature of the other motion gesture, *Tilt Device,* suggests either a use for simple actions like a binary +/- rating (making use of the left-right or front-back movements of *Tilt Device)*, or for any navigation function along two or three axes. *Tilt Device* is not applied in our test application, because the application does not use binary ratings.

3.3 Test Application User Interface

In the test application, the user can use a search interface to select among movie genres and find items. The search interface can be reached from the start screen, main menu or through the options menu. After searching, a list of corresponding items is shown (Fig. 2, left). Users can scroll up and down the list, remove items from the list or select an item to display more details by using *Single Tap*. The item details screen (Fig. 2, right) shows information for the selected movie and allows for bookmarking and rating the item. In addition, an options menu is available on every screen to return to the search screen or main menu of the application (Fig. 2, right). The following functions are available and implemented by at least two input options each:

- *Bookmark*: The user can bookmark an item by using on-screen or options menu buttons (Fig. 2, right), or by using the *Double Tap* gesture in the item details screen
- *Find Random Item*: Accessible application-wide through the options menu or by using the *Shake Device* gesture
- *Save Search Parameters*: This function is available in the search screen via an on-screen button or by a *Double Tap* in this screen
- *Find Similar*: The item details screen shows three movies similar to the selected one ("similar to this

[1] On most systems, a dedicated software or hardware button opens up the options menu

movie" part in Fig. 2, right). The user has the option to find more similar items by using an on-screen button or the *Flick* gesture

- *Exclude Item*: Available in the list view as an on-screen button (Fig. 2, left) or via the *Flick* gesture
- *Rate Item*: Users can rate items in the item details screen by selecting the "Rate" on-screen button (Fig. 2, right). Then, a rating scale of 1 to 10 stars appears. The user can set his or her desired rating by either using the rating scale as an on-screen button or applying the *One-Finger-Hold Pinch* (cf. Section 2.1) gesture.

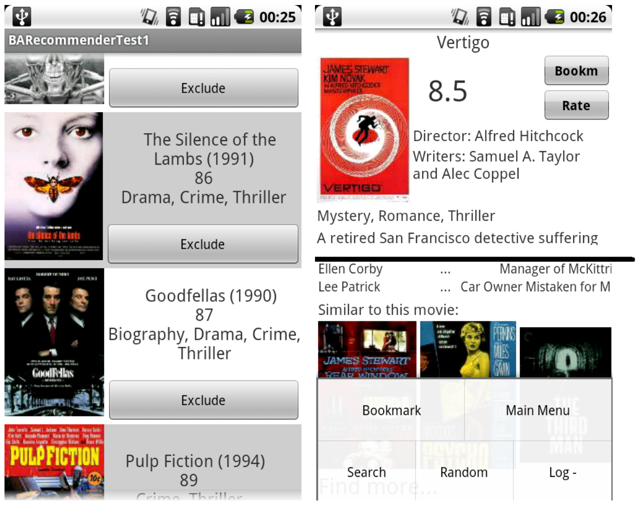

00:25
BARecommenderTest1
Exclude
The Silence of the Lambs (1991)
86
Drama, Crime, Thriller
Exclude
Goodfellas (1990)
87
Biography, Drama, Crime, Thriller
Exclude
Pulp Fiction (1994)
89
00:26
Vertigo
Bookm
Rate
8.5
Director: Alfred Hitchcock
Writers: Samuel A. Taylor and Alec Coppel
Mystery, Romance, Thriller
A retired San Francisco detective suffering
Ellen Corby ... Manager of McKittri
Lee Patrick ... Car Owner Mistaken for M
Similar to this movie:
Bookmark
Main Menu
Search
Random
Log -
Find more...

Figure 2. List of recommendations (left). Item details with options menu (right).

4. USER STUDY

4.1 Study Setup and Methodology

We have conducted a user study to find out what input method for a given function is preferred by the test users. The evaluation was performed with each of the participants individually. To start, each user was given an explanation of the application and was then allowed to practice navigating the different functions and input methods for about ten minutes. The participants then had to perform a set of 18 instructions in the application in a certain order. The list mentioned the required tasks only; the input method to perform them was not specified. By doing so, we tested which input method the test persons found more intuitive to use for a certain task. The beginning of the sequence of instructions read as follows: (1) *Find Random Item*, (2) *Find Similar Item*, (3) *Rate Item*, (4) *Open Main Menu*, (5) *Open My Recommendations*, (6) *Exclude Item from Recommendations*, and so on. Some of the requested functions appeared several times in the list, for example *Find Random Item* was requested three times. This was used to test whether participants would change their preferred input method for a particular function during the experiment.

We recorded every user action in a log file. After a test user completed the scenario, he or she had to fill out a survey concerning his or her opinions about the input methods for the requested instructions and about the handling of the gestures in particular.

4.2 Log File Analysis

16 persons with mixed backgrounds participated in the study. Other than a few users skipping a few tasks from the instruction list, all subjects completed the given scenario. We first analyzed the log file to understand which input options the users chose to complete a given task.

Out of a total of 44 recorded usages, the *Find Random* function was initiated 26 times using the *Shake Device* gesture, and 18 times using the options menu button (see Fig. 2, right). This represents a 59.1% usage rate for the implemented gesture. Interestingly, only one out of the 16 users elected to use both available input methods; every other user exclusively used either the gesture or the button for the three instances of *Find Random* in our instruction list.

The *Bookmark Item* function is represented three times in the scenario. The users chose to use the *Double Tap* gesture 27 out of 46 times (58.7%). However, at one instance in the scenario, the activity in focus is the item list, which only implements bookmarking via double tapping. In this case, 11 of 16 users (68.8%) chose the *Double Tap* gesture, while the rest of the users elected to take additional time to first open an item's details page and bookmark there. While the users were on an item's details page, they called only 16 of 35 (45.7%) instances of *Bookmark Item* using the *Double Tap* gesture. All differences to 100% in this paragraph are due to the uses of the on-screen bookmark button – the options menu button was never used.

The use of the *Save Search Parameters* function was requested only once in the scenario and can be called using *Double Tap* or an on-screen button. This is the function with the clearest favorite among the input methods: 15 out of 16 users (93.8%) chose the on-screen button.

The scenario contained two instances of the *Exclude Recommended Item* function, operable via Flick gesture or an on-screen button. 18 of 32 (56.3%) calls were made using gestural interaction. A relatively high number of users used both input methods for this task: 4 out of 16 participants (25%). This is even though the two instances of the *Exclude Recommended Item* task occurred directly after each other in our task list.

Rate Item and *Find Similar Item* each occur two times in the scenario. For both, a clear preference towards the standard input method of an on-screen button can be seen: for *Rate Item*, only 10 of 32 instances (31.3%) were operated with the *One-Finger-Hold Pinch* gesture. Even more one-sided, the *Find Similar Item* function was only initiated using *Flick* in 3 of 32 cases (9.4%). The remaining percentages represent instances of functions called via on-screen button.

4.3 Survey Results

In the first part of the survey we asked the participants how intuitive they find the input methods for the six functions on a scale from 1 to 5. Figure 3 illustrates the results with a higher number meaning "more intuitive". In general, the results correspond to the log file very well: input methods that were actually preferred and used by the participants received higher grades for intuitivity. For example, the participants find the on-screen buttons for *Save Search* and *Find Similar* very intuitive. On the other hand, the *Shake Device* for *Find Random Item*, *Double Tap* for *Bookmark* and *Flick* for *Exclude Item* gestures received higher grades in comparison with on-screen or option menu buttons.

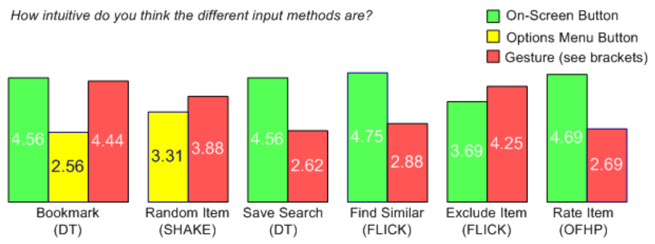


Figure 3. Average of users' ratings how intuitive each function's input method was.

The next question was whether inclusion of an on-screen button was worth the necessary screen space for it. Our users mostly were in favor of it: the majority of users denied this question for *Exclude Item* only (Fig. 4). Interestingly, this is the only on-screen button in the list view (Fig. 2).

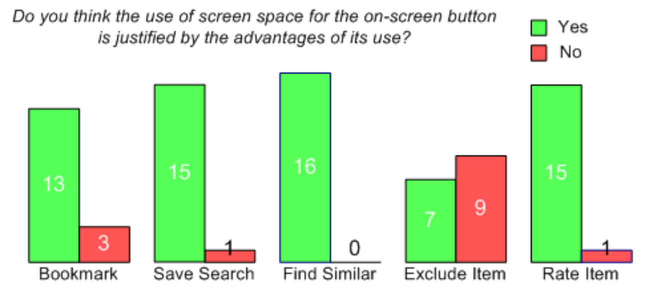


Figure 4. Screen space usage for on-screen buttons

The goal of the next part of the survey was to determine the user's favorite input method for each function. The distribution of choices for each function is shown in Fig. 5 and is comparable to the grades for intuitivity: interaction patterns that users perceived as intuitive were chosen as favorite input method.

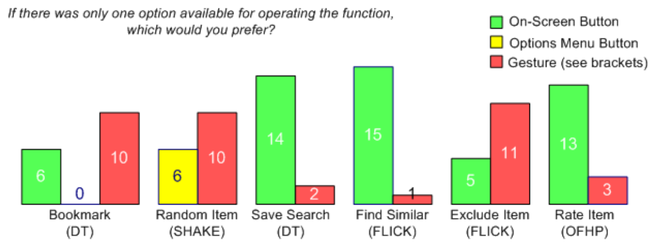


Figure 5. Selecting only one input option for each function

We also asked the test users about their prior experience with touchscreen devices and analyzed whether it would relate to differences in the results. The most significant difference was that 62.5% of the users with more prior touchscreen experience rated the *Shake Device* gesture as intuitive, while only 12.5% did so among the users with less experience. We noted a similar difference regarding the *Flick* gesture.

Concerning the ease of handling of the four gestures, the participants considered all gestures, except *One-Finger-Hold Pinch (OFHP),* as easy to handle in general. One of the problems with *OFHP* was that lifting a finger while adjusting the desired rating for item ends the rating process. In addition, the calibration for the rating scale of one to ten stars was difficult. So this gesture might be more suitable for simpler tasks with fewer options.

5. DISCUSSION AND CONCLUSION

The results of the study presented in this work may be used to improve the design of user interfaces for mobile recommender systems and other similar applications. Our study showed that users preferred the simpler, easier to handle gestures over the more complex ones. Complex gestures like *One-Finger-Hold Pinch* must be carefully calibrated for ease of handling. Omitting on-screen buttons is only an option in activities where content space is rare, in our case the overview list of items. For the item detail screen, simply touching a button was the favorite input method most of the times. The options menu was not very popular in any of the used cases. This is likely due to the fact that opening the options menu is an extra effort that users do not tend to make when other input methods are available.

While *Double Tap* for bookmarking items was received very well, the *Double Tap* gesture for *Save Search Parameters* was not very popular and received low grades for intuitivity. This may be due to the layout of the corresponding screens because users might have the fear of accidently tapping on other interface elements. In essence, the use of gestural interaction patterns seems to depend on the actual screen and function detail. Interestingly, users did not change their preferred input mode much during the test: they mostly used the same method for the same task throughout the scenario. Users with more experience with touchscreen devices were more open towards gestures than users with less experience.

Future work includes studying in more detail how more complex gestures can be introduced in mobile recommender systems to improve user interaction. Moreover, a long-term study would be interesting because user acceptance might change if smartphone users get more and more used to complex motion gestures.

6. REFERENCES

[1] Konstan, J.A., and Riedl, J. 2012. Recommender systems: from algorithms to user experience. *User Model. User-Adapt. Interact.* 22, 1-2 (April 2012), 101-123. DOI= http://dx.doi.org/10.1007/s11257-011-9112-x.

[2] Ricci, F. 2011. Mobile recommender systems. *J. of IT & Tourism.* 12, 3 (April 2010), 205-231. DOI= http://dx.doi.org/10.3727/109830511X12978702284390.

[3] Saffer, D. 2008. *Designing Gestural Interfaces.* O'Reilly, Sebastopol.

[4] Schulze, F., Woerndl, W., and Ludwig, M. 2012. A gesture-based interface for map-based exploratory search on tablets. In *Proc. Mobility and Web Behavior Workshop, MobileHCI '12 Conference* (San Francisco, CA, September 21 – 24, 2012).

[5] Cho, S.J., Murray-Smith, R. and Kim, Y.B. 2007. Multi-context photo browsing on mobile devices based on tilt dynamics. In *Proc. of the 9th International Conference on Human Computer Interaction with Mobile Devices and Services* (Singapore, September 09 – 12, 2007). MobileHCI '07. ACM, New York, NY, 190-197. DOI= http://doi.acm.org/10.1145/1377999.1378006.

[6] Negulescu, M., Ruiz, J., Li, Y., and Lank, E. 2012. Tap, swipe, or move: attentional demands for distracted smartphone input. In *Proc. of the Int. Working Conference on Advanced Visual Interfaces* (Capri Island, Italy, May 21 – 25, 2012). AVI '12. ACM, New York, NY, 173-180. DOI= http://doi.acm.org/10.1145/2254556.2254589.

[7] Rico, J., and Brewster, S. 2010. Usable gestures for mobile interfaces: evaluating social acceptability. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems* (Atlanta, GA, April 10 – 15, 2010). CHI '10. ACM, New York, NY, 887-896. DOI= http://doi.acm.org/10.1145/1753326.1753458.

The Role of Emotions in Context-aware Recommendation

Yong Zheng, Robin Burke, Bamshad Mobasher
Center for Web Intelligence
School of Computing, DePaul University
Chicago, Illinois, USA
{yzheng8, rburke, mobasher}@cs.depaul.edu

ABSTRACT

Context-aware recommender systems try to adapt to users' preferences across different contexts and have been proven to provide better predictive performance in a number of domains. Emotion is one of the most popular contextual variables, but few researchers have explored how emotions take effect in recommendations – especially the usage of the emotional variables other than the effectiveness alone. In this paper, we explore the role of emotions in context-aware recommendation algorithms. More specifically, we evaluate two types of popular context-aware recommendation algorithms – context-aware splitting approaches and differential context modeling. We examine predictive performance, and also explore the usage of emotions to discover how emotional features interact with those context-aware recommendation algorithms in the recommendation process.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information filtering

General Terms

Algorithm, Experiment, Performance

Keywords

Recommendation, Context, Context-aware recommendation, Emotion, Affective recommender system

1. INTRODUCTION

Advances in affective computing have enabled recommender systems to take advantage of emotions and personality, leading to the development of affective recommender systems (ARS) [18]. At the same time, the emerging technique of context-aware recommender systems (CARS) takes contexts into consideration, converting a two-dimensional matrix of ratings organized by user and item: *Users* $\times$ *Items* $\rightarrow$ *Ratings*, into a multidimensional rating space [1]. CARS have been demonstrated to be effective in a variety of applications and domains [4, 15, 10, 24, 13]. Emotional variables are often included as contexts in CARS, which enables the further development of both ARS and CARS. Typical emotional contexts in recommender system domain are the ones relevant to users' subject moods or feelings, for example, user mood when listening to music tracks (e.g. happy, sad, aggressive, relaxed, etc) [19, 9], or users' emotions after seeing a movie (e.g. user may feel sad after seeing a tragic movie) [13].

Emotional context was first exploited by Gonzalez *et al* [8] for the recommender system domain. This work was followed by others [9, 18, 13, 17] considering emotions as contexts in CARS research. While the effectiveness of emotions as contextual variables is therefore well-established, there is little research that has examined specifically the role that these emotional variables play. We define "the role of emotions" as the concerns from two aspects – whether emotions are useful or effective to improve recommendation performance? And, the usage of emotions in the recommendation process – how emotions are used in the recommendation algorithms, e.g. which emotional variables are selected? which algorithm components are they applied to? and so forth. Currently, most research are focused on demonstrating the effectiveness of emotional variables, and few research further explore the usage of the emotions in the recommendation process.

In this paper, we explore the role of emotions by two classes of popular context-aware recommendation algorithms – context-aware splitting and differential context modeling. Since both of these approaches require that the algorithm learn the importance and utility of different contextual features, they help reveal how emotions work in each algorithm and what roles they can play in recommendation.

The purpose of this study is therefore to address the following research questions:

1. *Emotional Effect*: Are emotions useful contextual variables for context-aware recommendation?
2. *Algorithm Comparison*: What algorithms are best suited to make use of emotional variables? Do they outperform the baseline algorithms?
3. *Usage of Contexts*: How do emotional variables compete with other contextual variables, such as location and time?
4. *Roles*: How can we understand the specific roles emotional variables can play in those context-aware recommendation algorithms?

RecSys'13, October 12–16, 2013, Hong Kong, China.
Paper presented at the 2013 Decisions@RecSys workshop in conjunction with the 7th ACM conference on Recommender Systems. Copyright ©2013 for the individual papers by the papers' authors. Copying permitted for private and academic purposes. This volume is published and copyrighted by its editors.

2. RELATED WORK

Gonzalez *et al* [8] explored emotional context in recommender systems in 2007. They pointed out that,"*emotions are crucial for user's decision making in recommendation process. The users always transmit their decisions together with emotions.*" With the rapid development of context-aware recommender systems, emotion turns out to be one of important and popular contexts in different kinds of domains, especially in the music and movie domain. For music recommendation, the emotional context is appealing because it can be used to establish a bridge between music items and items from other different domains, and perform cross-media recommendations [6, 1, 9]. Movie recommendation is another domain where emotion turns out to be popular in recent years. In 2010, Yue *et al* [16] produced the overall winner in a recent challenge on context-aware movie recommendation by mining mood-specific movie similarity with matrix factorization. More recent research [18, 13] motivates the tendency of taking emotions as contexts to assist contextual recommendations. Research has demonstrated that emotions can be influential contextual variables in making recommendations, but few of them explore how emotions interact with recommendation algorithm – the usage of emotional variables in the recommendation process.

As described in [1], there are three basic approaches to develop context-aware recommendation algorithms: pre-filtering, post-filtering, and contextual modeling. A pre-filtering approach applies a context-dependent criterion to the list of items, selecting those appropriate to a given context. Only the filtered items are considered for recommendation. A post-filtering approach is similar but applies the filter after recommendations have been computed. Contextual modeling takes contextual considerations into the recommendation algorithm itself. In this paper, we explore context-aware splitting approaches (a class of pre-filtering algorithms), and differential context modeling (a contextual modeling approach.)

The remainder of this paper is organized as follows. In Section 3 and Section 4, we formally introduce the two types of recommendation algorithms we are studying, including their capability to capture the role of emotional contexts in recommendation process. Sections 5 and 6 discuss the experimental evaluations and mining the role of emotions through those context-aware recommendation algorithms, followed by the conclusions and future work in Section 7.

3. CONTEXT-AWARE SPLITTING APPROACHES

Contextual pre-filtering is popular as it is straightforward to implement, and can be applied with most recommendation techniques. *Item splitting* [4, 5] is considered one of the most efficient pre-filtering algorithms and it has been well developed in recent research. The underlying idea of item splitting is that the nature of an item, from the user's point of view, may change in different contextual conditions, hence it may be useful to consider it as two different items [5]. *User splitting* [2, 15] is based on a similar intuition – it may be useful to consider one user as two different users, if he or she demonstrates significantly different preferences across contexts. In this section, we introduce those two approaches and also propose a new splitting approach – *UI splitting*, a simple combination of item and user splitting.

To better understand and represent the splitting approaches, consider the following movie recommendation example:

Table 1: Movie Ratings in Contexts

User	Item	Rating	Time	Location	Companion
U1	T1	3	Weekend	Home	Friend
U1	T1	5	Weekend	Cinema	Girlfriend
U1	T1	?	Weekday	Home	Family

In Table 1, there are one user $U1$, one item $T1$ and two ratings (the first two rows) in the training data and one unknown rating that we are trying to predict (the third row). There are three contextual dimensions – time (weekend or weekday), location (at home or cinema) and companion (friend, girlfriend or family). In the following discussion, we use *contextual dimension* to denote the contextual variable, e.g. "Location" in this example. The term *contextual condition* refers to a specific value in a contextual dimension, e.g. "home" and "cinema" are two contextual conditions for "Location".

3.1 Item Splitting

Item splitting tries to find a contextual condition on which to split each item. The split should be performed once the algorithm identifies a contextual condition in which items are rated significantly differently. In the movie example above, there are three contextual conditions in the dimension *companion*: friend, girlfriend and family. Correspondingly, there are three possible alternative conditions: "*friend and not friend*", "*girlfriend and not girlfriend*", "*family and not family*". Impurity criteria [4] are used to determine whether and how much items were rated differently in these alternative conditions, for example a t-test or other statistical metric can be used to evaluate if the means differ significantly across conditions.

Item splitting iterates over all contextual conditions in each context dimension and evaluates the splits based on the impurity criteria. It finds the best split for each item in the rating matrix and then items are split into two new ones, where contexts are eliminated from the original matrix – it transforms the original multi-dimensional rating matrix to a 2D matrix as a result. Assume that the best contextual condition to split item T1 in Table 1 is "Location = *home and not home*", T1 can be split into T11 (movie T1 being seen at home) and T12 (movie T1 being seen not at home). Once the best split has been identified, the rating matrix can be transformed as shown by Table 2(a).

This example shows a *simple split*, in which a single contextual condition is used to split the item. It is also possible to perform a *complex split* using multiple conditions across multiple context dimensions. However, as discussed in [5], there are significant costs of sparsity and potential overfitting when using multiple conditions. We use only simple splitting in this work.

Table 2: Transformed Rating Matrix

(a) by Item Splitting

User	Item	Rating
U1	T11	3
U1	T12	5
U1	T11	?

(b) by User Splitting

User	Item	Rating
U12	T1	3
U12	T1	5
U11	T1	?

(c) by UI Splitting

User	Item	Rating
U12	T11	3
U12	T12	5
U11	T11	?

3.2 User Splitting

Similarly, user splitting tries to split users instead of items. It can be easily derived from item splitting as introduced above. Similar impurity criteria can also be used for user splits. Assume that the best split for user $U1$ in Table 1 is "Companion = *family and not family*", $U1$ can be split into $U11$ ($U1$ saw the movie with family) and $U12$ ($U1$ saw the movie with others). The rating matrix can be transformed as shown by Table 2(b). The first two rows contain the same user $U12$ because $U1$ saw this movie with others (i.e. not family) rather than family as shown in the original rating matrix.

3.3 UI Splitting

UI splitting is a new approach proposed in this paper – it applies item splitting and user splitting together. Assuming that the best split for item and user splitting are the same as described above, the rating matrix based on UI splitting can be shown as Table 2(c). Here we see that both users and items were transformed, creating new users and new items.

Thus, given n items, m users, k contextual dimensions and d distinct conditions for each dimension, the time complexity for those three splitting approaches is the same as $O(nmkd)$ [5]. The process in UI splitting is simply an application of item splitting followed by user splitting on the resulting output. We keep the contextual information in the matrix after the transformation by item splitting so that user splitting can be performed afterwards.

3.4 Role of Emotions in Splitting

Context-aware splitting approaches have been demonstrated to improve predictive performance in previous research, but few of those research results report on the details of the splitting process. More specifically, it is useful to know which contextual dimensions and conditions are selected, and the statistics related to those selections. For our purposes, it is possible to explore how emotions interact with the splitting process by the usage of contexts in the splitting process, which may help discover the role of emotions from this perspective. In this paper, we try all three context-aware splitting approaches, empirically compare their predictive performance, and also explore the distributions of the usage of contexts (emotional variables included) for splitting operations.

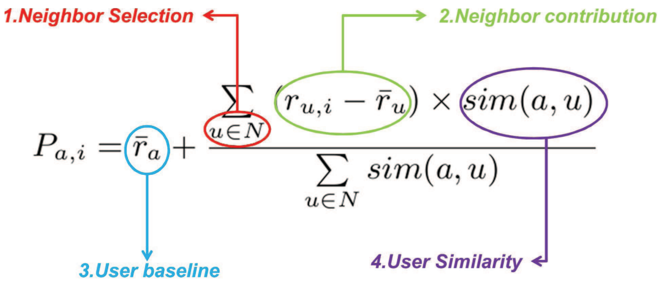
4. DIFFERENTIAL CONTEXT MODELING

Differential context modeling (DCM) is a general contextual recommendation framework proposed in [23]. It can be applied to any recommendation algorithm. The "*differential*" part of the technique tries to break down a recommendation algorithm into different functional components to which contextual constraints can be applied. The contextual effect for each component is maximized, and the joint effects of all components contribute the best performance for the whole algorithm. The "*modeling*" part is focused on how to model the contextual constraints. There are two approaches: *context relaxation* and *context weighting*, where context relaxation uses an optimal subset of contextual dimensions, and context weighting assigns different weights to each contextual factor. Accordingly, we have two approaches in DCM: *differential context relaxation* (DCR) [21, 22, 20] and *differential context weighting* (DCW) [24].

4.1 DCM in UBCF

We have successfully applied DCM to user-based collaborative filtering (UBCF), item-based collaborative filtering (IBCF) and Slope-One recommendation algorithms in our previous research [23], showing that DCM is an efficient approach to improve context-aware prediction accuracy. However, we did not explore much about the modeling part in our previous work; that is, how contexts are selected, relaxed or weighted.

Recall that we separate different functional components from the recommendation algorithm, therefore, how the algorithms relax or weigh contextual variables can tell the role of the contexts in different components. In this paper, we continue to apply DCM to UBCF, where the four components we separate in UBCF can be described as follows:



$$P_{a,i} = \bar{r}_a + \frac{\sum_{u \in N} (r_{u,i} - \bar{r}_u) \times sim(a,u)}{\sum_{u \in N} sim(a,u)}$$

Figure 1: Algorithm Components in UBCF

Figure 1 shows the original rating prediction by the well-known Resnick's UBCF algorithm [14], where a is a user, i is an item, and N is a neighborhood of K users similar to a. The algorithm calculates $P_{a,i}$, which is the predicted rating that user a is expected to assign to item i. Then we decompose it to four components:

Neighborhood selection UBCF applies the well-known k nearest-neighbor (kNN) approach, where we can select the top-k neighbors from users who have rated on the same item i. If contexts are taken into consideration, the neighborhood can be further restricted so that users have also rated the item i in the *same contexts*. This gives a context-specific recommendation computation. However, the strict application of such a filter greatly increases the sparsity associated with user comparisons. There may only be a small number of cases in which recommendations can be made. DCM offers two potential solutions to this problem. DCR searches for the optimal relaxation of the context, generalizing the set of contextual features and contextual conditions to reduce sparsity. DCW uses weighting to increase the influence of neighbors in similar contexts.

Neighbor contribution The neighbor contribution is the difference between a neighbor's rating on an item i and his or her average rating over all items. Context relaxation and context weighting can be applied to the computation of $\bar{r}_u$. This computation is replaced by one in which this average is computed over ratings from a relaxed set of contexts in DCR, or the average rating can be aggregated by a weighted average across similar contexts under DCW.

User baseline The computation of $\bar{r}_a$ is similar to the neighbor's average rating and can be made context-dependent in the same way.

User similarity The computation of neighbor similarity $sim(a,u)$ involves identifying ratings $r_{u,i}$ and $r_{a,i}$ where the users have rated items in common. For context-aware recommendation, we can additionally add contextual constraints to this part: use ratings given in matching contexts with context relaxation or ratings weighted by contextual similarity using context weighting.

With these considerations in mind, we can derive a new rating prediction formula by applying DCR or DCW to UBCF. More details about the prediction equations and technical specifications can be found in [24].

4.2 The Role of Emotions in DCM

One advantage of DCM is that it allows us to explore the role of contexts in each algorithm component. DCM seeks to optimize the contribution of context in each component, and so, the output of the optimization phase, in which contextual dimensions are selected and/or weighted, can reveal the relative importance of different contextual features in different algorithm components. In this paper, we provide comparisons and also explore the role of emotions in these two classes of context-aware recommendation algorithms.

5. EXPERIMENTAL SETUP

In this section, we introduce the data sets, evaluation protocols and the specific configurations in our experiments.

5.1 Data Sets

We examine context-aware splitting approaches and DCM with the real-world data set *LDOS-CoMoDa*, which is a movie dataset collected from surveys and used by Odic *et al* [12, 13, 17]. After filtering out subjects with less than 5 ratings and rating records with incomplete feature information, we got the final data sets containing 113 users, 1186 items, 2094 ratings, 12 contextual dimensions and the rating scale is 1 to 5. We created five folds for cross validation purposes and all algorithms are evaluated on the same five folds. Specific descriptions of all the contextual dimensions and conditions can be found in previous work [12] and the description of the data set online [1].

In our experiments, we use all 12 contextual dimensions, including three emotional contexts: *Mood*, *DominantEmo* and *EndEmo*, and non-emotional contexts, including *Location*, *time*, etc. The goal is to compare emotional contexts with others. For example, in DCR, only influential contextual variables for a specific component are selected – whether emotional contexts are selected or not can indicate the significance of their impacts for this component. For a comprehensive comparison, we also performed experiments without emotional contexts to better evaluate the importance of the emotional dimensions.

In this data, there are the three contextual dimensions that contain emotional information. "EndEmo" is the emotional state experienced at the end of the movie. "DominantEmo" is the emotional state experienced the most during watching, i.e. what emotion was induced most times during watching. "Mood" is what mood the user was in during that part of the day when the user watched the movie. Mood has lower maximum frequency than emotions, it changes slowly, so we assumed that it does not change during watching. "EndEmo" and "DominantEmo" contain the same seven conditions: *Sad, Happy, Scared, Surprised, Angry, Disgusted, Neutral*, where "Mood" only has simple three conditions: *Positive, Neutral, Negative*.

5.2 Configurations

To evaluate the performance of context-aware splitting approaches, we used four splitting criteria described in [5]: t_{mean}, t_{chi}, t_{prop} and t_{IG}. t_{mean} estimates the statistical significance of the difference in the means of ratings associated to each alternative contextual condition using a t-test. t_{chi} and t_{prop} estimates the statistical significance of the difference between two proportions – high ratings (>R) and low ratings ($\leq$R) by chi square test and z-test respectively, where we choose R = 3 as in [5]. t_{IG} measures the information gain given by a split to the knowledge of the item i rating classes which are the same two proportions as above.

[1]http://212.235.187.145/spletnastran/raziskave/um/comoda/comoda.php

Usually, a threshold for the splitting criteria should be set so that users or items are only be split when the criteria meets the significance requirement. We use an arbitrary value of 0.2 in the t_{IG} case. For t_{mean}, t_{chi} and t_{prop}, we use 0.05 as the p-value threshold. A finer-grained operation is to set another threshold for each impurity value and each data set. We deem it as a significant split once the p-value is no larger than 0.05. We rank all significant splits by the impurity value, and we choose the top first (highest impurity) as the best split. Items or users without qualified splitting criteria are left unchanged.

In DCM, we used the same configuration in our previous work [24]: we select the top-10 neighbors in UBCF, choose 100 as the maximal iteration in the optimization process, and Pearson Correlation is used as the user similarity measure. See our previous work for more details.

5.3 Evaluation Protocols

For the evaluation of predictive performance, we choose the root mean square error (RMSE) evaluated using the 5-fold cross validation. The data set is relatively small and some subjects in the survey were required to rate specific movies, thus precision and recall may be not a good metric, but we plan to evaluate them and other metrics in our future work.

We applied DCM only to user-based collaborative filtering. For splitting approaches, there are more options – we evaluate the performance of three recommendation algorithms: user-based collaborative filtering (UBCF), item-based collaborative filtering (IBCF) and matrix factorization (MF) for each splitting approach. Koren [11] introduced three MF techniques: MF with rating bias (BiasMF), Asymmetric SVD (AsySVD) and SVD++; we found that BiasMF was the best choice for this data.

We used the open source recommendation engine MyMediaLite v3.07 [7] to evaluate UBCF, IBCF and BiasMF in our experiments. (We choose K=30 for KNN-based UBCF and IBCF.) In order to better evaluate MF techniques, we tried a range of different factors ($5 \leq N \leq 60$, increment 5) and training iteration T ($10 \leq T \leq 100$, increment 10). Other parameters like learning and regularization factors are handled by MyMediaLite, where stochastic gradient descent is used as the optimization method.

For comparison purposes, we choose the well-known context-aware matrix factorization algorithm (CAMF) [3], i.e. CAMF_C, CAMF_CI and CAMF_CU [2] as the baseline. We tried our best to fine-tune the configurations (e.g. learning parameters, training iterations, etc) for CAMF in order to make a reasonable and comprehensive comparison.

6. EXPERIMENTAL RESULTS

In this section, the comparisons of predictive performance are introduced first, followed by the discussion of emotional roles discovered in our experiments.

6.1 Prediction

We use three treatments of the context information – one is the data with *All Contexts* where both emotional contexts and non-emotional variables are included, and the second one is the data omitting the emotional context dimensions marked by *No Emotions* in the table below. The third one is the data with *Emotions Only* emitting all non-emotional variables. The overall experimental results are shown in Table 3, where the numbers in underlined in italic are the best RMSEs by each approach (i.e. the best one in

[2]CAMF_CU is a CAMF approach which utilizes the interaction between contexts and users. [12]

Table 3: Overall Comparison of RMSE
(The results for splitting approaches are based on BiasMF.)

	Algorithms	**All Contexts**	**No Emotions**	**Emotions Only**
CAMF [3]	CAMF_C	1.012	1.066	*0.968*
	CAMF_CI	1.032	1.083	*1.019*
	CAMF_CU	0.932	1.021	*0.902*
DCM [24]	DCR	*1.043*	1.057	1.046
	DCW	*1.017*	1.037	1.036
Splitting Approaches	Item Splitting	*1.011*	1.014	1.014
	User Splitting	*0.913*	0.971	0.932
	UI Splitting	**0.892**	**0.942**	**0.903**

each row), and the bold numbers are the best RMSEs by each data forms (i.e. the best one in each column).

The comparison of performances of those approaches over the three forms of data can be visualized by Figure 2. We see that including emotions as contexts can improve RMSE compared with the situation we only use non-emotional contexts (i.e. *No Emotions*). The results differ when we switch our attention to the "*Emotions Only*" one. In CAMF, it helps achieve the lowest RMSE with *Emotions Only* data. But, including all of the contextual information yielded the best RMSE for the other approaches: DCM and context-aware splitting algorithms.

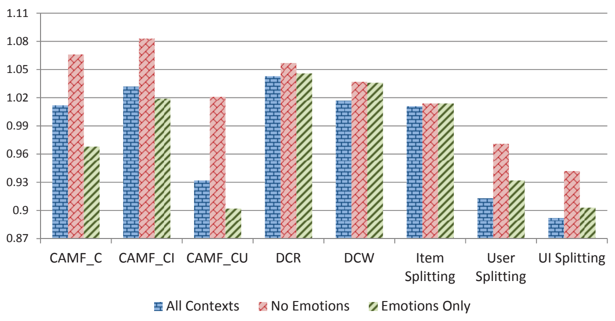


Figure 2: Comparison of RMSE Over Three Contextual Situations

This result is not surprising because both DCM and splitting approaches choose among contextual features, deciding which to apply in recommendation. CAMF, on the other hand, uses all available context information in performing its factorization. Without the benefit of feature selection, adding additional features to CAMF may increase noise.

From an overall view, UI splitting has the best RMSE across all data treatments among those context-aware recommendation algorithms. Table 4 shows more details of the predictive performance among the three splitting approaches if we include emotions as the contexts (i.e. using all contexts). The numbers are shown as RMSE values, where the numbers in bold are the best performing RMSE values for each recommendation algorithm across all three splitting approaches. The numbers in underlined in italic are the best RMSE values achieved for the data set using each splitting approach. The numbers in underlined in bold are the global best RMSE for the data set.

These tables show that adding emotions to contexts is able to provide improvement in terms of RMSE, thus answering research question 1 in the affirmative. It also suggests an answer to question 2: that the UI splitting approach outperforms other two splitting approaches if it is configured optimally. In particular, the best RMSE values are achieved by UI splitting using MF as the recommendation algorithm with t_{chi} as the splitting criteria.

Table 4: Comparison of Predictive Performance (RMSE) Among Splitting Approaches

		LDOS-CoMoDa			
	Algorithms	t_{mean}	t_{chi}	t_{prop}	t_{IG}
Item Splitting	UBCF	1.040	1.021	1.028	1.043
	IBCF	1.030	1.024	1.026	1.034
	BiasMF	1.020	*1.011*	1.016	1.020
User Splitting	UBCF	1.026	0.987	0.999	1.052
	IBCF	0.985	0.967	0.985	1.039
	BiasMF	0.934	*0.913*	0.928	1.011
UI Splitting	UBCF	**1.012**	**0.956**	**0.989**	**1.042**
	IBCF	**0.972**	**0.946**	**0.972**	**1.020**
	BiasMF	**0.927**	**0.892**	**0.915**	**0.998**

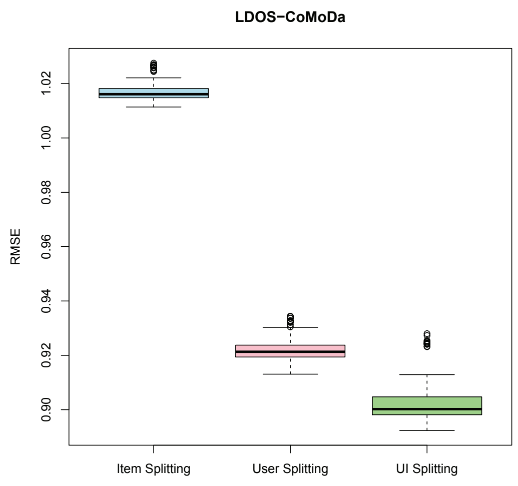


Figure 3: Boxplot of RMSE Among Splitting Approaches

Generally, MF is the best performing recommendation algorithm. This is not surprising because splitting increases sparsity and MF approaches are designed to handle sparsity data. Because the difference in RMSE is small, we show the boxplot of RMSEs among the best performing item splitting, user splitting and UI splitting approaches (i.e. the configuration as the underlined values in Table 4) in Figure 3. The data in the figure are the 120 RMSE values which comes from the training iterations – different factors

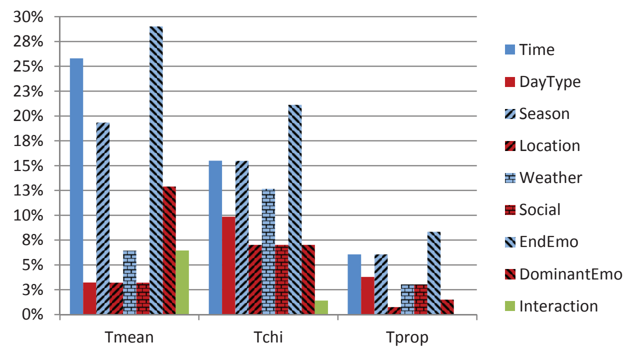


Figure 4: Item Splitting

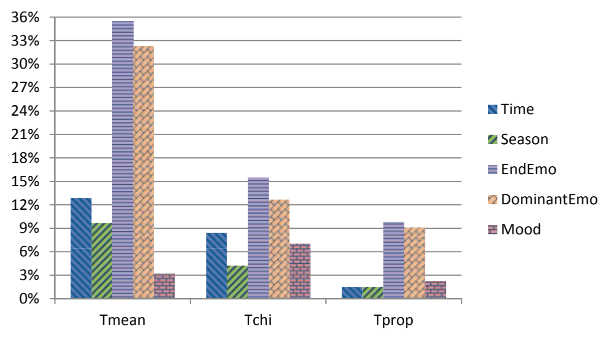


Figure 5: User Splitting

Table 5: Context Relaxation and Weighting by DCM

Algorithm Components	**Context Relaxation By DCR**	**Context Weighting By DCW**
Neighbor Selection	N/A	Day, *Mood*
Neighbor Contribution	Movie Year, Genre	Movie Genre
User Baseline	*DominantEmo, EndEmo*, Movie Language	*DominantEmo, EndEmo*, Interaction
User Similarity	*EndEmo*, Location	*DominantEmo*

($5 \leq N \leq 60$, with 5 increment in each step) and training iteration T ($10 \leq T \leq 100$, with 10 increment in each step). The figure confirms the comparative effectiveness of UI splitting – the box is significantly lower than the other two approaches with the same training iterations.

6.2 The Role of Emotions

As mentioned before, one reason we sought to explore the usage of emotional contexts to discover how emotions interact within context-aware recommendation algorithms. Both splitting and DCM offer insights into how contextual dimensions and features are influential for recommendation.

6.2.1 *Context-aware Splitting*

In splitting approaches, the splitting statistics help discover how contexts were applied in our experiments. In Figure 4 and 5, we show the top selected contextual dimensions for item splitting and user splitting[3]. The right legend indicates the contextual dimensions, and the y axis denotes the percentage of splits (item splits or user splits) using each dimension. For a clearer representation in the figures, we only show contextual dimensions used on more than 5% (item splitting) or 6% (user splitting) of the recommendations. We do not show results of t_{IG} because this splitting criterion had the worst performance.

In general, the top two dimensions are consistent across those three impurity criteria: *EndEmo* and *Time* for item splitting and *EndEmo* and *DominantEmo* for user splitting. However, the percentages as y axis in the figures are different, not to mention that the selected condition in each dimension differs too, which results in different performance by using various impurity criteria. In terms of the specific selected emotional conditions, the results are not consistent – the top context dimension for item and user splitting is the same – *EndEmo*, but the most frequent selected condition in this dimension is "Happy" for item splitting and "Neutral" for user splitting.

[3]The actual splitting is based on a specific contextual condition in a dimension, but the results of selected contextual conditions are fuzzy, thus the distribution of selected contextual dimensions is explored and reported here.

EndEmo denotes the emotion of the users after seeing the movie, and this result is consistent with previous work [12] on this data. Obviously, emotion is a personal quality and can be considered as more dependent with users other than items – we conjecture that this may be the underlying clue explaining why user splitting works better than item splitting for the LDOS-CoMoDa data. And in user splitting, the top two selected contextual dimensions are the two emotional variables: *EndEmo* and *DominantEmo* – emotions are generated and owned by users, therefore they are more dependent with users other than items. This pattern is confirmed by the comparison between two CAMF approaches: CAMF_CI and CAMF_CU – contexts are more dependent with users than with items, which results in better performances by CAMF_CU.

In short, the statistics based on splitting approaches reveal the importance of emotions – at least the top first selected contextual dimension is the "EndEmo" for both item splitting and user splitting.

6.2.2 *Differential Context Modeling*

In DCM, the context selection and context weighting can be examined and show, in a detailed way, the contribution of each contextual dimension in the final optimized algorithm. The results are shown in Table 5, where the four components in UBCF were described in the previous section. "N/A" in the table indicates no contextual constrains were placed. For clearer representation, we did not show specific weights of contexts in DCW; instead, we only list variables which were assigned weights above a threshold of 0.7. The weights are normalized to 1, and 0.7 therefore represents a very influential dimension.

We can see in the table that emotional variables are selected in DCR and weighted significantly in DCW and for which components. Emotion is influential for a specific component but may not for other ones. In DCR for example, *EndEmo* turns out to be influential when measuring user similarities and user baselines, but it is not that significant in computing the neighbor contribution.

It is not surprising that the results from DCW are not fully consistent with ones from DCR. DCW is a finer-grained approach. Emotional variables are assigned to neighbor selection in DCW but not for the same component in DCR, and the specific selected emotions

Table 6: Comparison of Context-aware Splitting and Differential Context Weighting

Answers	**Context-aware Splitting Approaches**	**Differential Context Weighting**
Q1. Emotional Effect	Yes. RMSE is improved with emotional context dimensions.	Yes. RMSE is improved with emotional context dimensions.
Q2. Algorithm Comparison	UI splitting is the best and outperforms DCM and CAMF approaches.	DCW is better than DCR and also works better than some CAMF approaches in specific contextual situations.
Q3. Usage of Contexts	Emotional dimension is the top selected dimension for context splitting.	Emotional dimensions get more weight than other contextual variables in most algorithm components.
Q4. Roles	*EndEmo* is used as the top first selected context for both user and item splitting. *DominantEmo* is the top second one in user splitting.	Emotions are significantly influential for some algorithm components, e.g. *EndEmo* and *DominantEmo* for user similarities and baselines in DCW.

are different too, e.g. it is *DominantEmo* selected in DCW for user similarity calculation, but it is the *EndEmo* in DCR. In DCW, the weights for *EndEmo* and *DominantEmo* are close to 1 (the weights are all above 0.92) in the component of user baseline, which implies significant emotional influence on this component. Emotional contexts are important in DCM, where it can be further confirmed by Table 3 – the RMSE values are increased if we remove emotions from the contexts.

The result by DCM is useful for further applications, such as affective computing or marketing purposes. Take the results of DCW in Table 5 for example, *Mood* is influential for selecting neighbors, which implies that if two users rated the same item under the same mood situation, it is highly possible that they are the good neighbor candidates for each other (though neighbor selection also depends on the user similarities). Similarly, *DominantEmo* is useful to measure user similarities, which infers that users rated items similarly with the same dominant emotions are probably the good neighbors in user-based collaborative filtering.

7. CONCLUSION AND FUTURE WORK

In conclusion, both context-aware splitting approaches and DCM are able to reveal how emotions interact with algorithms to improve recommendation performance by exploring the usage of contexts in the recommendation or splitting process. More specifically, in context-aware splitting approaches, the percentage of emotional contexts used by item splits or user splits can tell the importance of emotions in distinguishing different user rating behavior. DCM provides a way to see very specifically which emotional contexts are influential for which components in the recommendation process.

Table 6 examines how our research questions are answered for each type of context-aware recommendation. As discussed above, we find that contextual dimensions keyed to emotions are very useful in recommendation (Question 1) and that UI splitting outperforms the DCM and CAMF approaches (Question 2).

Question 3 asks about how emotional dimensions compare with other contextual information. Our results show that emotion-linked context makes an important contribution to context-aware recommendation, although other dimensions are also certainly important. For example, we see that *EndEmo* is the top selected contextual dimension in item and user splitting, with *Time* running the second in item splitting, and *DominantEmo* is the top second selected contextual dimension in user splitting.

Our fourth research question asks about how those two classes of context-aware recommendation algorithms can tell the roles of emotions by usage of contexts. Context-aware approaches are able to infer the roles by the distribution of usage of emotions selected for the splitting process. DCM techniques help answer this question, by pointing out which components make good use of different contextual variables – showing here that *DominantEmo* is important for calculating the user's baseline. Note that the neighbor contribution component in DCW is alone in making significant use of non-emotional context dimensions. This is interesting because in this component we are determining which movies to use to compute the neighbor's baseline for prediction. It appears that the system prefers to use non-emotional considerations in making use of others' ratings, even though the user's baseline is computed based on emotional dimensions.

In addition, we can get extra information from our splitting experiments because splitting is performed based on specific contextual conditions. As described above, we found in particular that "Happy" vs not-"Happy" was the important split on the *EndEmo* dimension for item splitting. Essentially, the algorithm is indicating that there is an important difference between users who feel happy at the conclusion of a given movie and those that do not.

In our future work, we plan to continue our exploration of these algorithms and more data using additional metrics, such as recall and/or normalized discounted cumulative gain. We are also looking at additional data sets to see if similar effects are found with respect to emotions, as well as the empirical comparison among those context-aware recommendation algorithms. We also plan to examine the effects by the correlations between different contextual variables, e.g. how emotional effects change if emotions are significantly dependent with other contexts or features. In addition, it is interesting to explore the association among emotions, user profiles, item features and users' ratings. For example, user may feel "*sad*" after seeing a tragedy movie, but the emotion could be "*happy*" because he or she saw such a good movie even if it is a tragedy. Therefore, which specific emotions will result in a higher rating? the "*sad*" or the "*happy*"?

8. REFERENCES

[1] G. Adomavicius, B. Mobasher, F. Ricci, and A. Tuzhilin. Context-aware recommender systems. *AI Magazine*, 32(3):67–80, 2011.

[2] L. Baltrunas and X. Amatriain. Towards time-dependant recommendation based on implicit feedback. In *ACM RecSys' 09, Proceedings of the 4th International Workshop on Context-Aware Recommender Systems (CARS 2009)*, 2009.

[3] L. Baltrunas, B. Ludwig, and F. Ricci. Matrix factorization techniques for context aware recommendation. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 301–304. ACM, 2011.

[4] L. Baltrunas and F. Ricci. Context-based splitting of item ratings in collaborative filtering. In *Proceedings of the third*

ACM conference on Recommender systems, pages 245–248. ACM, 2009.

[5] L. Baltrunas and F. Ricci. Experimental evaluation of context-dependent collaborative filtering using item splitting. *User Modeling and User-Adapted Interaction*, pages 1–28, 2013.

[6] S. Berkovsky, T. Kuflik, and F. Ricci. Cross-domain mediation in collaborative filtering. In *User Modeling 2007*, pages 355–359. Springer, 2007.

[7] Z. Gantner, S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Mymedialite: A free recommender system library. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 305–308. ACM, 2011.

[8] G. Gonzalez, J. L. de la Rosa, M. Montaner, and S. Delfin. Embedding emotional context in recommender systems. In *Data Engineering Workshop, 2007 IEEE 23rd International Conference on*, pages 845–852. IEEE, 2007.

[9] B.-j. Han, S. Rho, S. Jun, and E. Hwang. Music emotion classification and context-based music recommendation. *Multimedia Tools and Applications*, 47(3):433–460, 2010.

[10] N. Hariri, B. Mobasher, R. Burke, and Y. Zheng. Context-aware recommendation based on review mining. In *IJCAI' 11, Proceedings of the 9th Workshop on Intelligent Techniques for Web Personalization and Recommender Systems (ITWP 2011)*, pages 30–36, 2011.

[11] Y. 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*, pages 426–434. ACM, 2008.

[12] A. Odic, M. Tkalcic, J. F. Tasic, and A. Košir. Relevant context in a movie recommender system: Users' opinion vs. statistical detection. In *ACM RecSys' 12, Proceedings of the 4th International Workshop on Context-Aware Recommender Systems (CARS 2012)*, 2012.

[13] A. Odić, M. Tkalčič, J. F. Tasič, and A. Košir. Predicting and detecting the relevant contextual information in a movie-recommender system. *Interacting with Computers*, 25(1):74–90, 2013.

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

[15] A. Said, E. W. De Luca, and S. Albayrak. Inferring contextual user profiles – improving recommender performance. In *ACM RecSys' 11, Proceedings of the 4th International Workshop on Context-Aware Recommender Systems (CARS 2011)*, 2011.

[16] Y. Shi, M. Larson, and A. Hanjalic. Mining mood-specific movie similarity with matrix factorization for context-aware recommendation. In *Proceedings of the Workshop on Context-Aware Movie Recommendation*, pages 34–40. ACM, 2010.

[17] M. Tkalčič, U. Burnik, A. Odić, A. Košir, and J. Tasič. Emotion-aware recommender systems–a framework and a case study. In *ICT Innovations 2012*, pages 141–150. Springer, 2013.

[18] M. Tkalcic, A. Kosir, and J. Tasic. Affective recommender systems: the role of emotions in recommender systems. In *ACM RecSys Workshop on Human Decision Making*, 2011.

[19] A. L. Uitdenbogerd and R. G. van Schyndel. A review of factors affecting music recommender success. In *ISMIR*, volume 2, pages 204–208, 2002.

[20] Y. Zheng, R. Burke, and B. Mobasher. Assist your decision-making in various situations: Differential context relaxation for context-aware recommendations. In *Research Colloquium, School of Computing, DePaul University, Chicago IL, USA*, 2012.

[21] Y. Zheng, R. Burke, and B. Mobasher. Differential context relaxation for context-aware travel recommendation. In *13th International Conference on Electronic Commerce and Web Technologies (EC-WEB 2012)*, pages 88–99, 2012.

[22] Y. Zheng, R. Burke, and B. Mobasher. Optimal feature selection for context-aware recommendation using differential relaxation. In *ACM RecSys' 12, Proceedings of the 4th International Workshop on Context-Aware Recommender Systems (CARS 2012)*. ACM, 2012.

[23] Y. Zheng, R. Burke, and B. Mobasher. Differential context modeling in collaborative filtering. In *School of Computing Research Symposium*. DePaul University, USA, 2013.

[24] Y. Zheng, R. Burke, and B. Mobasher. Recommendation with differential context weighting. In *The 21st Conference on User Modeling, Adaptation and Personalization (UMAP 2013)*, pages 152–164, 2013.

Managing Irrelevant Contextual Categories in a Movie Recommender System

Ante Odić[*]
University of Ljubljana, Faculty of Electrical Engineering
Tržaška cesta 25
Ljubljana, Slovenia
ante.odic@ldos.fe.uni-lj.si

Marko Tkalčič
Johannes Kepler University
Department for Computational Perception
Altenberger Str. 69
Linz,Austria
marko.tkalcic@jku.at

Andrej Košir
University of Ljubljana, Faculty of Electrical Engineering
Tržaška cesta 25
Ljubljana, Slovenia
andrej.kosir@ldos.fe.uni-lj.si

ABSTRACT

Since the users' decision making depends on the situation the user is in, contextual information has shown to improve the recommendation procedure in context-aware recommender systems (RS). In our previous work we have shown that relevant contextual factors have significantly improved the quality of rating prediction in RS, while the irrelevant ones have degraded the prediction. In this work we focus on the detection of relevant contextual conditions (i.e., values of contextual factors) which influence the users' decision making process. The goals are (i) to lower the intrusion for the end user by simplifying the acquisition process, and (ii) to reduce the sparsity of the acquired data during the contextual modeling. The results showed significant improvement in the rating prediction task, when managing the irrelevant contextual conditions by the approach that we propose in this paper.

Keywords

context-aware, recommender systems, user modeling

1. INTRODUCTION

Over the past decade, employing contextual information in recommender systems (RS) has been a popular research topic. Contextual information is defined as the information that can be used to describe the situation and the environment of the entities involved in such systems [5]. Since users' decision making depends on the situation the user is in, contextual information has shown to improve the recommendation results in context-aware recommender systems (CARS) [1, 3, 10], as well as other personalized services [11].

In this work we follow the terminology described in [4]: *contextual factor* refers to a specific type of contextual information (e.g. weather), *contextual condition* refers to a specific value for a contextual factor (e.g. sunny), and *contextual situation* refers to a specific set of these contextual conditions that describe the context in which the user consumed the item.

In our previous work [10] we have proposed a methodology for detecting the relevancy of contextual factors, and have shown that relevant contextual factors significantly improved the quality of rating prediction in RS, while the irrelevant ones degraded the prediction. Similar results were achieved in [3] by assessing the relevancy of contextual factors.

In this work we focus on the detection of relevant contextual conditions, i.e., the values of contextual factors, which influence the users' decision making process, with the goal of lowering the intrusion for the end user by simplifying the acquisition process, and to reduce the sparsity of the acquired data during the contextual modeling.

1.1 The Problems of Many Contextual Conditions: Sparsity and Acquisition

One of the main problems with contextual factors with many contextual conditions is the sparsity of rating data. For example, let us say a specific user rated 20 items in different contextual situations. For uncontextualized modeling that would be a fair amount of ratings from that specific user. However, let us say some contextual factor contains ten contextual conditions and users ratings are equally distributed across those conditions. That would mean that for each condition we only have two ratings from that user. For this reason it would be better to have a lower number of contextual conditions per contextual factor.

In addition, since the contextual data is often explicitly acquired through questionnaires (e.g. in [2] or [10]), lowering the number of questions and possible conditions shortens the questionnaire. This is important for lowering the amount of time required from users to provide ratings and the associated context.

To summarize, the acquisition and usage of contextual factors with many contextual conditions has two negative sides:

- questionnaire size (effort required from a user)
- sparsity (ratings are distributed in many categories)

Therefore, it would be beneficial to reduce the number of

[*]Corresponding author.

RecSys'13, October 12–16, 2013, Hong Kong, China.
Paper presented at the 2013 Decisions@RecSys workshop in conjunction with the 7th ACM conference on Recommender Systems.

contextual conditions of the relevant contextual factors.

1.2 Problem Statement

The problem with the reduction of the number of contextual condition is how to select the conditions to remove and how to merge the contextual conditions in order to reduce their number.

By avoiding the relevant contextual conditions we might lose valuable information. Hence, we need to detect irrelevant conditions, identify how they should be merged and handled during the acquisition, and during the training and the preparation of recommendations.

In this article we propose an approach by which we achieve the following goals:

- we identify contextual conditions which should be avoided or merged in questionnaires
- we manage irrelevant categories during training to utilize provided ratings and decrease the sparsity

In the following sections we describe the approach, dataset used and the experimental results.

1.3 Experimental Design

In this subsection we describe the experimental design used in this study. For each contextual variable available in the dataset we do the following steps in order to manage irrelevant categories.

First we do the **contextual-condition-relevancy detection**. At this stage we use statistical testing in order to detect which contextual conditions of a specific contextual factor are irrelevant and do not have the impact on the ratings. We consider a contextual condition to be relevant if the users' behavior (how users rate items) is different for that condition than for other conditions. If the users do not rate items differently for that contextual condition than otherwise, we consider the condition to be irrelevant.

The next step is to determine whether these irrelevant conditions could be merged with the relevant ones. For example, if *rainy weather* would be detected as irrelevant, but *cloudy weather* as relevant, perhaps they could be merged into a combined category *cloudy/rainy weather*. Hence, we call this step the **context-categories-merging determination**. Once the merging possibilities are determined, we may use them for two separate tasks: (i) improving the questionnaire, and (ii) improving the contextualized model of users decisions.

Improving the questionnaire. If in a system, after a sufficient amount of data was collected, it is determined that several contextual-factors' conditions are irrelevant and could be merged with others, the questionnaire used for the data acquisition should be modified. (Similarly, if the data is being collected implicitly through sensors, the acquisition procedure should be modified). In this way the number of questions in the questionnaire could be reduced and thus the time required from users to fill-in these questionnaires would be reduced. However, it might be the case that the merges are too complex to employ them in the questionnaire as we will show in the following sections.

Improving the model. In addition to improving the questionnaire, merges should be employed in the model as well. By using the irrelevant conditions in the model during training, the rating data is being used to train the contextualized parameters which depend on the irrelevant contextual conditions. Instead these ratings should be used for training the parameters that depend on the relevant contextual conditions. Hence, by merging categories we are able to use the rating data for the more meaningful task (which consequently reduces the sparsity of ratings), which would result in a better trained model. We will evaluate this task by comparing the root mean square error (RMSE) of the rating prediction with and without merging of context categories, and the results form random merging of contextual conditions as a baseline.

Figure 1 shows the whole procedure described in the article.

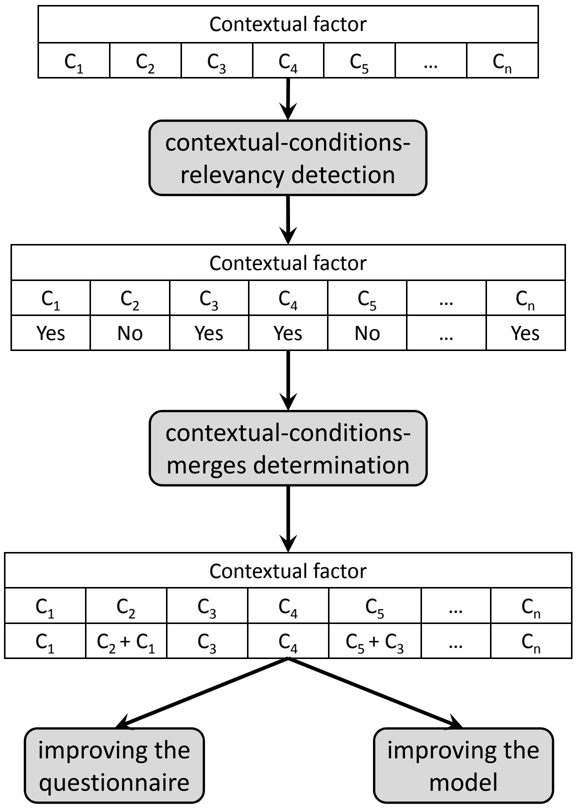

Contextual factor
C_1 C_2 C_3 C_4 C_5 ... C_n
contextual-conditions-relevancy detection
Contextual factor
C_1 C_2 C_3 C_4 C_5 ... C_n
Yes No Yes Yes No ... Yes
contextual-conditions-merges determination
Contextual factor
C_1 C_2 C_3 C_4 C_5 ... C_n
C_1 $C_2 + C_1$ C_3 C_4 $C_5 + C_3$ ... C_n
improving the questionnaire
improving the model

Figure 1: Experimental design. For each contextual variable, the relevancy of categories is detected, merging possibilities are determined and used to improve both the data acquisition and the modeling procedure.

2. MATERIALS AND METHODS

In this section we describe the dataset used in this study and describe each step of the experimental design in more details.

2.1 Dataset

For the purposes of this work we have used the *Context Movie Dataset* (LDOS-CoMoDa), that we have acquired in our previous work [10].

We have created an online application for rating movies which users are using in order to track the movies they watched and obtain the recommendations (`www.ldos.si/recommender.html`). Users are instructed to log into the system after watching a movie, enter a rating for a movie and fill in a simple questionnaire created to explicitly acquire the contextual information describing the situation during the consumption.

The part of the dataset used in this study consists of 1611 ratings from 89 users to 946 items with 12 associated contextual factors. Additional information about our *Context Movies Database* (LDOS-CoMoDa) can be found in [9] and [10].

All the contextual factors and conditions acquired are listed in Table 1.

Table 1: Contextual factors in the LDOS-CoMoDa dataset.

Contextual variable	Description
time	morning, afternoon, evening, night
daytype	working day, weekend, holiday
season	spring, summer, autumn, winter
location	home, public place, friend's house
weather	sunny/clear, rainy, stormy, snowy, cloudy
social	alone, partner, friends, colleagues, parents, public, family
endEmo	sad, happy, scared, surprised, angry, disgusted, neutral
dominantEmo	sad, happy, scared, surprised, angry, disgusted, neutral
mood	positive, neutral, negative
physical	healthy, ill
decision	user's choice, given by other
interaction	first, n-th

2.2 Contextual-Condition-Relevancy Detection

In order to determine if a contextual condition of a specific contextual factor is relevant, we use the **Wilcoxon rank-sum test** in the following way. For each condition (e.g., sunny weather) of a specific contextual factor (e.g., weather), we observe two populations of ratings: ratings associated with that condition only (e.g., sunny weather), and ratings associated with any other condition of the same contextual factor (e.g.,rainy, cloudy, snowy and stormy). We use the **Wilcoxon rank-sum test** to compare these two populations. More specifically, we test the null hypothesis that the ratings from these two populations are sampled from a continuous distributions with equal medians. If we reject the null hypothesis, the medians are different, which means that the users tend to rate items differently during the tested condition (e.g., sunny) compared to the other conditions (e.g. rainy, cloudy, snowy and stormy). If this is the case, we determine that the tested contextual condition is relevant. Otherwise we determine that since there is no difference in ratings, such condition has no impact and is thus irrelevant. The Wilcoxon rank-sum test was chosen over the t-test because the compared samples were not normally distributed.

The described approach was done on the population level, i.e., on the data from the whole population and not for each user separately. Hence, contextual conditions are detected as relevant or irrelevant with regards the whole population of users.

2.3 Contextual-Condition-Merges Determination

Once the irrelevant conditions of each contextual factor are detected, we proceed to merge them with relevant categories. In order to determine which categories should be merged, we compare the distribution of ratings for each irrelevant condition with the distribution for each relevant condition separately (e.g. sunny vs. rainy, sunny vs. cloudy, sunny vs. snowy and sunny vs. stormy). Once again, this is tested with the **Wilcoxon rank-sum test**. In this case, if the test determined that the medians of the ratings distributions for the irrelevant and relevant conditions are equal, we determine that these conditions can be merged. This is because there is no difference in rating when users were in these two separate conditions.

However, the proposed methodology might yield a type of error during merges. It might occur that we determine the two conditions could be merged when in fact they should not. This exception might occur if the distributions were similar, yet, for different user-item pairs ratings were drastically different on different contextual conditions. This is an open issue we plan to address in the future work.

2.4 Merging Contextual Conditions

Once we determine which contextual condition should be merged we implement merging in two separate tasks: (i) improving the questionnaire and (ii) improving the model.

2.4.1 Merging in Questionnaires

In our system we acquire the contextual information explicitly through questionnaire. Hence, we implement merging in the questionnaire by modifying the list of possible contextual conditions users choose from. For example, in our system we have the contextual factor *season*, which contains the following contextual conditions: *spring*, *summer*, *autumn*, *winter*. Let us say that we have determined *summer* to be an irrelevant condition and that it should be merged with the relevant condition *autumn*. We would simply change possible answers in the questionnaire into: *spring*, *summer/autumn*, *winter*. In this way we lower the amount of possible answers, and stop associating ratings with irrelevant contextual condition. Of course, if possible, a new name for the combined condition could be used in the questionnaire.

If contextual information would be acquired implicitly through sensors, merging would be implemented in the step of processing sensor data into contextual conditions.

2.4.2 Merging during Modeling

In this study we used the contextualized matrix factorization algorithm for modeling the interaction between the users and the movie items. Matrix factorization (MF) is a latent-factor model that is widely used in RS ([8, 3, 6, 7]). We implement the contextualization by making users' rating biases context dependent as in [10].

The contextualized users' biases with the matrix factorization **(CUB-MF)** approach uses the contextual information for the contextualized users' biases. Only the users' biases are context dependent. This approach follows the idea that the users' rating behaviour is different on different occasions. The matrix factorization in CUB-MF was made using the following equation:

$$\hat{r}(u,h) = \mu + b_h + b_u(c) + \vec{q}_h^{\,T} \cdot \vec{p}_u, \quad (1)$$

where $\hat{r}(u,h)$ is the predicted rating for user u and item h, $\vec{q}_h$ is the item's latent-feature vector, $\vec{p}_u$ is the user's latent-feature vector. The user's bias b_u and the item's bias b_h measure the deviations of the user's u and the item's h ratings from the rating average μ.

To inspect the impact of merging contextual conditions of contextual factors on the rating prediction, we trained our model for each contextual factor separately, i.e. using only a single contextual factor at the time.

The standard way, of training (without merging) the contextualized model is done in the following way: the algorithm loops through all the ratings in the training set, and calculates the prediction error $e(u,h,c) = r(u,h,c) - \hat{r}(u,h,c)$ for each predicted rating $\hat{r}(u,h,c)$ and real rating $r(u,h,c)$, for user u, item h and contextual condition c. Among other uncontextualized parameters, we modify the contextualized user's u bias by the equation:

$$b_u(c) \leftarrow b_u(c) + \gamma \cdot (e(u,h,c) - \lambda \cdot b_u(c)). \quad (2)$$

Hence, if the contextual condition was, for example *summer*, we would update $b_u(sunny)$.

When we implement merging during modeling, for each calculated error of prediction, we update the contextualized parameters of all merged conditions, if such exist. Therefore, if, for example, the contextual condition *summer* has to be merged with the condition *autumn*, we would use $e(u,i,summer)$ to update $b_u(sunny)$ and $b_u(autumn)$ simultaneously. In this way we reduce the negative impact of sparsity by utilizing ratings associated with irrelevant conditions to train parameters contextualized by the relevant ones. In addition, during training, for each calculated error of prediction, we also train the uncontextualized users' biases. Once the model is trained, on the testing set, the uncontextualized users' biases are used to predict the ratings associated with the irrelevant contextual conditions. In this way, the algorithm simply avoids the contextualized rating prediction in the case of the irrelevant contextual condition.

2.5 Random Merging as a Baseline

In order to test the positive impact of our procedure for detecting irrelevant contextual conditions, and determining merges, it is important to compare the results from our approach with the fair baseline. It could be that the improvement in the rating prediction is not due to our merging technique, but due to any type of merging simply because we lower the sparsity. In another words, it is important to test if we would get equally improved results by randomly merging several conditions.

Therefore we have implemented a random merging method in the following way: for every contextual factor we count the exact number of irrelevant conditions and determined merges, and select the same amount of random conditions and random merges. In this way we replicate the same amount of merges but select the conditions to be merged randomly.

The results for our approach and the random merges are achieved on 10 different folds.

3. RESULTS

In the cases of the *time*, *daytype* and *location* contextual factors, all conditions were found irrelevant, hence no merges are possible. In the cases of the *decision*, *interaction*, and *physical* contextual factors, all conditions were found relevant, hence no merges are needed. For the remaining contextual factors, table 2 contains the results of the contextual-condition-relevancy detection, and merges determination.

The figures 2, 3, 4, 5, 6 and 7 show the results of the matrix factorization rating prediction.

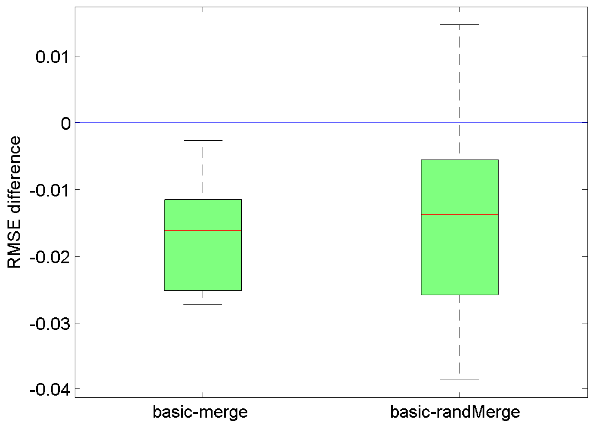


Figure 2: Rating prediction results for dominant emotion.

On each figure, boxplots are presented: one from our merging method (*merge*) and the second one from the random merge baseline (*randMerge*). Both boxplots represent the RMSE difference between the basic model without merging (*basic*), and the *merge* and *randMerge* approaches. Therefore, if the result is above zero, the merging approach performed better (lower RMSE) than the basic approach without merging.

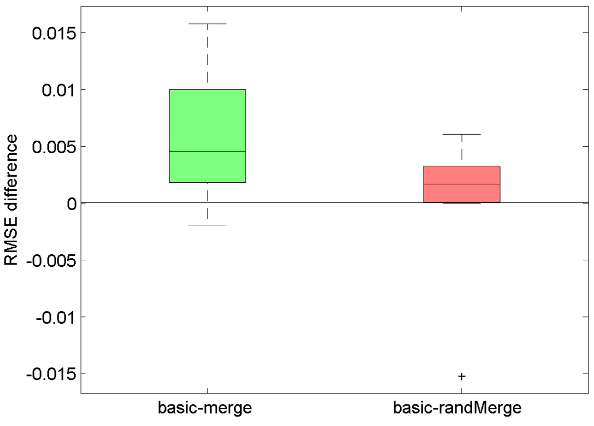


Figure 3: Rating prediction results for end emotion.

The **Wilcoxon signed-rank test** was used to test the statistical significance of the differences between basic and merging approaches. If the difference was statistically significant the box plot is colored green, otherwise it is colored red.

Table 2: Results of the contextual-condition-relevancy detection, and merges determination.

SEASON		
condition	**relevancy**	**merges**
spring	yes	
summer	no	autumn
autumn	yes	
winter	yes	

WEATHER		
condition	**relevancy**	**merges**
sunny	no	
rainy	no	
stormy	no	snowy
snowy	yes	
cloudy	no	

SOCIAL		
condition	**relevancy**	**merges**
alone	yes	
partner	yes	
friends	no	alone partner family
colleagues	no	alone partner family
parents	no	alone family
public	no	alone partner family
family	yes	

END EMOTION		
condition	**relevancy**	**merges**
sad	yes	
happy	yes	
fear	no	sad happy surprised
surprised	yes	
angry	yes	
disgusted	yes	
neutral	yes	

DOMINANT EMOTION		
condition	**relevancy**	**merges**
sad	yes	
happy	yes	
fear	no	sad happy surprised
surprised	yes	
angry	no	neutral
disgusted	yes	
neutral	yes	

MOOD		
condition	**relevancy**	**merges**
positive	yes	
neutral	yes	
negative	no	neutral

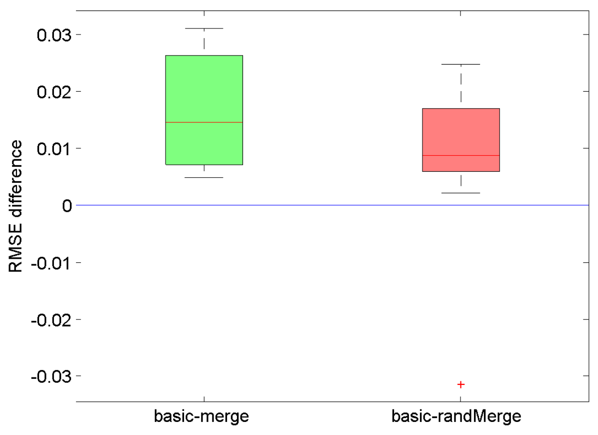


Figure 4: Rating prediction results for mood.

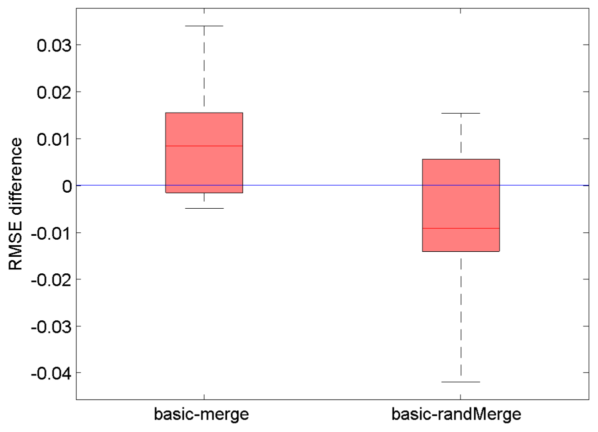


Figure 5: Rating prediction results for season.

3.1 Discussion

In the previous section we could observe different results for different contextual factors. It is interesting to note that contextual factors for which all the contextual conditions were detected as irrelevant (*time*, *daytype* and *location*) are those that were detected irrelevant themselves in our previous work [10]. Similarly, the contextual factors for which all the contextual conditions were detected as relevant (*decision*, *interaction*, and *physical*) are those that were detected as relevant themselves in our previous work. Therefore, we might conclude that such contextual factor for which all the contextual conditions are detected as irrelevant, can be observed as irrelevant and left out from the contextualized modeling altogether. For the remaining contextual factors we summarize the results in Table 3.

Implementing merges in questionnaire can be easily achieved for *season*, *weather* and *mood*, by simply merging conditions between possible answers. However, for *social* contextual condition, as it is shown in Table 2, there are conflicts which prevent us for merging. For example, the condition *parents* can be merged with *alone* and *family*, but not with *partner*, as it is the case with the conditions *friends*, *colleagues* and *public*.

Furthermore, for *end emotion* and *dominant emotion*, the irrelevant condition *fear* can be merged with multiple condi-

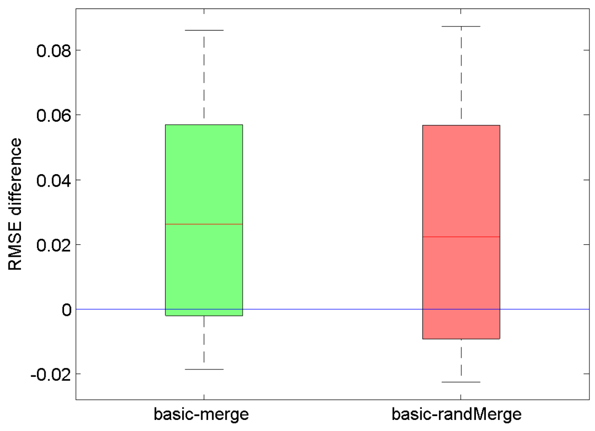


Figure 6: Rating prediction results for social.

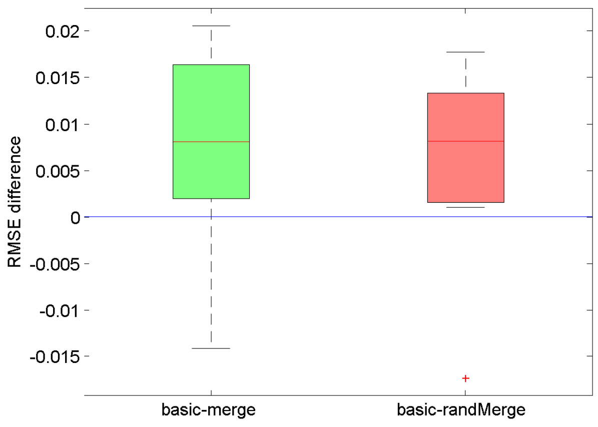


Figure 7: Rating prediction results for weather.

tions (*sad*, *happy*, *surprised*), however each of them is relevant and should be used alone as it is. Therefore, an opened issue remains - how such cases should be handled in questionnaires.

By implementing the proposed procedure for the detection of irrelevant contextual categories, and the proposed way to manage merges during modeling, we achieved significantly better results than without merging for the contextual factors *weather*, *social*, *end emotion* and *mood*. For the contextual factor *season* we achieved an improvement, however it was not statistically significant (Figure 5). In each case our procedure outperformed random-merging baseline, which did not lead to significantly improved results in any case. However, even in the case of random merging there is tendency towards better results with fewer conditions which confirms our assumption from the introduction: many contextual conditions have a large impact on the sparsity of ratings in the contextualized models.

The only contextual factor for which we observed unexpected results is the *dominant emotion*. In this case we achieved significantly worse results for both our approach and the random-merging baseline. We believe that this is an interesting open issue that we plan to address further in the future.

Table 3: Summary of the results. The table tells whether there is an improvement in the questionnaire or in the model, for each contextual factor separately.

	improvement	
context	**questionnaire**	**rating prediction**
season	yes	no
weather	yes	yes
social	no	yes
endEmo	?	yes
domEmo	?	no
mood	yes	yes

4. CONCLUSION AND FUTURE WORK

In this paper we proposed a procedure for detecting the relevancy of contextual conditions and how to manage such conditions by merging them with relevant ones. We implemented merging of contextual conditions on the questionnaire for acquiring contextual data, and into contextualized modeling based on matrix factorization. The results showed significantly improved results by our method, except in the case of one specific contextual factor.

For the future work we plan on researching further why anomalies can occur and how to predict and avoid them. Also, we are interested in solving conflicts described in this paper regarding the implementation of merges in questionnaires.

5. REFERENCES

[1] G. Adomavicius and A. 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.

[2] L. Baltrunas, B. Ludwig, S. Peer, and F. Ricci. Context-Aware Places of Interest Recommendations for Mobile Users. *Design, User Experience, and Usability. Theory, Methods, Tools and Practice*, pages 531–540, 2011.

[3] L. Baltrunas, B. Ludwig, S. Peer, and F. Ricci. Context relevance assessment and exploitation in mobile recommender systems. *Personal and Ubiquitous Computing*, pages 1–20, June 2011.

[4] V. Codina, F. Ricci, and L. Ceccaroni. Semantically-enhanced pre-filtering for context-aware recommender systems. In *Proceedings of the 3rd Workshop on Context-awareness in Retrieval and Recommendation*, pages 15–18. ACM, 2013.

[5] A. Dey and G. Abowd. Towards a better understanding of context and context-awareness. *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pages 304–307, 1999.

[6] Z. Gantner, S. Rendle, and L. Schmidt-Thieme. Factorization Models for Context- / Time-Aware Movie Recommendations Encoding Time as Context. In *Proceedings of the Workshop on Context-Aware Movie Recommendation*, pages 14–19, 2010.

[7] B. Hidasi and D. Tikk. Enhancing matrix factorization through initialization for implicit

feedback databases. In *Proceedings of the 2nd Workshop on Context-awareness in Retrieval and Recommendation*, CaRR '12, pages 2–9, New York, NY, USA, 2012. ACM.

[8] Y. Koren. Factorization Meets the Neighborhood : a Multifaceted Collaborative Filtering Model. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. 2008.

[9] A. Košir, A. Odić, M. Kunaver, M. Tkalcic, and J. Tasic. Database for contextual personalization. *ELEKTROTEHNIŠKI VESTNIK*, 78(5):270–274, 2011.

[10] A. Odić, M. Tkalčič, J. F. Tasič, and A. Košir. Predicting and detecting the relevant contextual information in a movie-recommender system. *Interacting with Computers*, 25(1):74–90, 2013.

[11] F. Toutain, A. Bouabdallah, R. Zemek, and C. Daloz. Interpersonal Context-Aware Communication Services. *IEEE Communications Magazine*, (January):68–74, 2011.

An Improved Data Aggregation Strategy for Group Recommendations

Toon De Pessemier
iMinds - Ghent University
G. Crommenlaan 8 box 201
B-9050 Ghent, Belgium
Toon.DePessemier@UGent.be

Simon Dooms
iMinds - Ghent University
G. Crommenlaan 8 box 201
B-9050 Ghent, Belgium
Simon.Dooms@UGent.be

Luc Martens
iMinds - Ghent University
G. Crommenlaan 8 box 201
B-9050 Ghent, Belgium
Luc1.Martens@UGent.be

ABSTRACT

Although most recommender systems make suggestions for individual users, in many circumstances the selected items (e.g., movies) are not intended for personal usage but rather for consumption in group. Group recommendations can assist a group of users in finding and selecting interesting items thereby considering the tastes of all group members. Traditionally, group recommendations are generated either by aggregating the group members' recommendations into a list of group recommendations or by aggregating the group members' preferences (as expressed by ratings) into a group model, which is then used to calculate group recommendations. This paper presents a new data aggregation strategy for generating group recommendations by combining the two existing aggregation strategies. The proposed aggregation strategy outperforms each individual strategy for different sizes of the group and in combination with various recommendation algorithms.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information Filtering; H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces

General Terms

Algorithms, Experimentation

Keywords

group recommendations, aggregation strategy, combining techniques

1. INTRODUCTION

Although the majority of the currently deployed recommender systems are designed to generate personal suggestions for individual users, in many cases content is selected and consumed by groups of users rather than by individuals. This strengthens the need for group recommendations, providing suggestions thereby considering the tastes of all group members. In the literature, group recommendations have mostly been generated by one of the following two data aggregation strategies [2].

The first aggregation strategy (aggregating recommendations) generates recommendations for each individual user using a general recommendation algorithm. Subsequently, the recommendation lists of all group members are aggregated into a group recommendation list, which (hopefully) satisfies all group members. Different approaches to aggregate the recommendation lists have been proposed during the last decade, such as least misery and plurality voting [7]. Most of them make a decision based on the algorithm's prediction score, i.e., a prediction of the user's rating score for the recommended item. One commonly used way to perform the aggregation is averaging the prediction scores of each member's recommendation list. The higher the average prediction score is, the better the match between the group's preferences and the recommended item.

The second grouping strategy (aggregating preferences) combines the users' preferences into group preferences. This way, the opinions and preferences of individual group members constitute a group preference model reflecting the interests of all members. Again, the members' preferences can be aggregated in different ways, e.g., by calculating the rating of the group as the average of the group members' ratings [7, 1]. After aggregating the members' preferences, the group's preference model is treated as a pseudo user in order to produce recommendations for the group using a traditional recommendation algorithm.

This paper presents a new data aggregation strategy, which combines the two existing strategies and outperforms each of them in terms of accuracy. For both individual data aggregation strategies, we used the average function to combine the individual preferences or recommendations. Although a switching scheme between both aggregation strategies has already been investigated [2], the proposed combined strategy is the first to generate group recommendations by using both aggregation strategies at once, thereby making a more informed decision.

2. EVALUATING GROUP RECOMMENDATIONS

A major issue in the domain of group recommender systems is the evaluation of the accuracy, i.e., comparing the generated recommendations for a group with the true preferences of the group. Performing online evaluations or interviewing groups can be partial solutions but are not feasible on a large scale or to extensively test alternative configurations. For example, in Section 5, five recommendation algorithms in combination with two data aggregation strategies are evaluated for twelve different group sizes, thereby leading to 120 different setups of the experiment. Therefore, we are forced to perform an offline evaluation, in which synthetic groups are sampled from the users of a traditional single-user data set. Since movies are often watched in group, we used the MovieLens (100K) data set for this evaluation.

In the literature, group recommendations have been evaluated several times by using a data set with simulated groups of users. Baltrunas et al. [1] used the MovieLens data set to simulate groups of different sizes (2, 3, 4, 8) and different degrees of similarity (high, random) with the aim of evaluating the effectiveness of group recommendations. Chen et al. [4] also used the MovieLens data set and simulated groups by randomly selecting the members of the group to evaluate their proposed group recommendation algorithm. They simulated group ratings by calculating a weighted average of the group members' ratings based on the users' opinion importance parameter. Quijano-Sánchez et al. [8] used synthetically generated data to simulate groups of people in order to test the accuracy of group recommendations for movies. In addition to this offline evaluation, they conducted an experiment with real users to validate the results obtained with the synthetic groups. One of the main conclusions of their study was that it is possible to realize trustworthy experiments with synthetic data, as the online user test confirmed the results of the experiment with synthetic data. This conclusion justifies the use of an offline evaluation with synthetic groups to evaluate the group recommendations in our experiment.

This offline evaluation is based on the traditional procedure of dividing the data set in two parts: the training set, which is used as input for the algorithm to generate the recommendations, and the test set, which is used to evaluate the recommendations. In this experiment, we ordered the ratings chronologically and assigned the oldest 60% to the training set and the most recent 40% to the test set, as this reflects a realistic scenario the best.

The used evaluation procedure was adopted from Baltrunas et al. [1] and is performed as follows. Firstly, synthetic groups are composed by selecting random users from the data set. All users are assigned to one group of a predefined size. Secondly, group recommendations are generated for each of these groups based on the ratings of the members in the training set. Since group recommendations are intended to be consumed in group and to suit simultaneously the preferences of all members of the group, all members receive the same recommendation list. Thirdly, since no group ratings are available, the recommendations are evaluated individually as in the classical single-user case, by comparing (the rankings of) the recommendations with (the rankings of) the items in the test set of the user using the Normalized Discounted Cumulative Gain (nDCG) at rank 5. The nDCG is a standard information retrieval measure, used to evaluate the recommendation lists [1].

3. RECOMMENDATION ALGORITHMS

The effectiveness of the different aggregation strategies is measured for different sizes of the group and in combination with various state-of-the art recommendation algorithms. The used implementation of **Collaborative Filtering** (CF) is based on the work of Breese et al [3]. This nearest neighbor CF uses the Pearson correlation metric for discovering similar users in the user-based approach (UBCF) or similar items in the item-based approach (IBCF) based on the rating behavior of the users. As **Content-Based recommender** (CB) the InterestLMS predictor of the open source implementation of the Duine framework [9] is adopted (and extended to consider extra metadata attributes). Based on the actors, directors, and genres of the content items and the user's ratings for these items, the recommender builds a profile model for every user. This profile contains an estimation of the user's preference for each genre, actor, and director that is assigned to a rated item, and is used to predict the user's preference for unseen media items by matching the metadata of the items with the user's profile. The used **hybrid recommender** (Hybrid) combines the recommendations with the highest prediction score of the IBCF and the CB recommender into a new recommendation list. The result is an alternating list of the best recommendations originating from these two algorithms. A user-centric evaluation comparing different algorithms based on various characteristics showed that this straightforward combination of CF and CB recommendations outperforms both individual algorithms on almost every qualitative metric [6]. As recommender based on matrix factorization, we opted for the open source implementation of the **SVD recommender** (SVD) of the Apache Mahout project [10]. This recommender is configured to use 19 features, which equals the number of genres in the MovieLens data set, and the number of iterations is set at 50. To compare the results of the various recommenders, the **popular recommender** was introduced as a baseline. This recommender generates for every user always the same list of most-popular items, which is based on the number of received ratings and the mean rating of each item.

4. COMBINING STRATEGIES

Previous research [5] has shown that the used aggregation strategy in combination with the recommendation algorithm has a major influence on the accuracy of the group recommendations. Certain algorithms (such as CB and UBCF) produce more accurate group recommendations when the aggregating preferences strategy is used, whereas other algorithms (such as IBCF and SVD) obtain a higher accuracy in combination with the aggregating recommendations strategy. So, the choice of the aggregation strategy is crucial for each algorithm in order to obtain the best group recommendations. Instead of selecting one individual aggregation strategy, traditional aggregation strategies can be combined with the aim of obtaining group recommendations which outperform the group recommendations of each individual aggregation strategy. In this context, Berkovsky and Freyne [2] witnessed that the aggregating recommendations strategy yields a lower MAE (Mean Absolute Error) than the aggregating preferences strategy if the user profiles have a low density (i.e., containing a low number of consumptions). In contrast for high-density profiles, the aggregating preferences strategy resulted in the lowest MAE, thereby outperforming the aggregating recommendations strategy in terms of accuracy. Therefore, Berkovsky and Freyne proposed a switching scheme based on the profile density, which yielded a small accuracy improvement compared to the individual strategies. However, their results were obtained in a very specific setting. They only considered the accuracy of recommendations generated by a CF algorithm, the MAE metric was used to estimate the accuracy, and they focused on the specific use case of recipe recommendations using a rather small data set (approximately 3300 ratings). Because of these specific settings, we were not able to obtain an accuracy improvement by using such a switching scheme on the MovieLens data set.

Therefore, we propose an advanced data aggregation strategy which combines both individual aggregation strategies thereby yielding an accuracy gain compared to each individual aggregation strategy for different recommendation algorithms. This combination of strategies aggregates the preferences of the users as well as their recommendations with the aim of merging the knowledge of the two aggregation strategies into a final group recommendation list. The idea is that if one of the aggregation strategies comes up with a less suitable or undesirable group recommendation, the other aggregation strategy can correct this mistake. This makes the group recommendations resulting from the combination of strategies more robust than the group recommendations based on a single aggregation strategy.

The two aggregation strategies are combined as follows. First, group recommendations are calculated by using the selected recommendation algorithm and the aggregating preferences strategy. The result is a list of all items, ordered according to their prediction score. In case of an individual aggregation strategy, the top-N items on that list are selected as suggestions for the group. After calculating the group recommendations using the aggregating preferences strategy, or in parallel with it, group recommendations are generated using the chosen algorithm and the aggregating recommendations strategy. Again, the result is an ordered list of items with their corresponding prediction score.

Both of these lists with group recommendation can still contain items that are less suitable for the group, even at the top of the list. The next phase will try to eliminate these items by comparing the two resulting recommendation lists. Items that are at the top of both lists are probably interesting recommendations, whereas items at the bottom of both lists are usually less suitable for the group. Less certainty exists about the items that are at the top of the recommendation list that is generated by one of the aggregation strategies but that are in the middle or even at the bottom of the recommendation list produced by using the other aggregation strategy. Therefore, both recommendation lists are adapted by eliminating these uncertain items in order to contain only items that appear at the top of both recommendation lists, thereby reducing the risk of recommending undesirable or less suitable items to the group. So, items that are ranked below a certain threshold position in (at least) one of the recommendation lists generated by the two aggregation strategies, are removed from both lists. If only one aggregation strategy is used, identifying uncertain items based on the results of a complementary recommendation list is not possible. In this experiment, we opted to exclude these items from the recommendation lists, that are not in the top-5% of both recommendation lists (i.e., the top-84 of recommended items for the MovieLens data set). As a result, the recommendation lists contains only items that are identified as 'the most suitable' by both aggregation strategies, ordered according to the prediction scores calculated using either the aggregating preferences strategy or the aggregating recommendations strategy.

Subsequently, the two recommendation lists are combined into one recommendation list by combining the prediction scores of each aggregation strategy per item. In this experiment, we opted for the average as method to combine the prediction scores. So in the resulting recommendation list, each item's prediction score is the average of the item's prediction score generated by the aggregating preferences strategy and the item's prediction score produced by the aggregating recommendations strategy. Alternative combining methods are also possible, e.g., a weighted average of the prediction scores with weights depending on the performance of each individual aggregation strategy. Then, the items are ordered by their new prediction score in order to obtain the final list of group recommendations.

5. RESULTS

Our combined aggregation strategy is compared to the individual aggregation strategies in Figure 1. Since users are randomly combined into groups and the accuracy of group recommendations is depending on the composition of the groups, the accuracy slightly varies for each partitioning of the users into groups. (Except for the partitioning of the users into groups of 1 member, which is only possible in 1 way.) Therefore, the process of composing groups by taking a random selection of users is repeated 30 times and just as much measurements of the accuracy are performed. So, the graph shows the mean accuracy of these measurements as an estimation of the quality of the group recommendations (on the vertical axis), as well as the 95% confidence interval of the mean value, in relation to the recommendation algorithm, aggregation strategy, and the group size. The group size is indicated on the horizonal axis. The vertical axis crosses the horizontal axis at the quality level of the most-popular recommender. The prefix "Combined" of the bar series stands for the proposed aggregation strategy which combines the aggregating preferences and aggregating recommendations strategy. The prefix "Pref" and "Rec" indicate the accuracy of the two individual strategies, respectively the aggregating preferences and aggregating recommendations strategy. For each algorithm, only the most accurate individual strategy is shown: aggregating preferences for UBCF and CB, aggregating recommendations for SVD, IBCF, and Hybrid [5].

The non-overlapping confidence intervals indicate a significant improvement of the combined aggregation strategy compared to the best individual aggregation strategy. Table 1 shows the results of the statistical T-tests comparing the mean accuracy of the recommendations generated by the best individual aggregation strategy and by the combined aggregation strategy for groups with size = 5. (Similar results are obtained for other group sizes.) The null hypothesis, H_0 = the mean accuracy of the recommendations generated by using the best individual aggregation strategy is equal to the mean accuracy of the recommendations generated by using the combined aggregation strategy. The small p-values (all smaller than 0.05) prove the significant accuracy improvement of our proposed aggregation strategy.

6. CONCLUSIONS

This paper presents a new strategy to aggregate the tastes of multiple users in order to generate group recommenda-

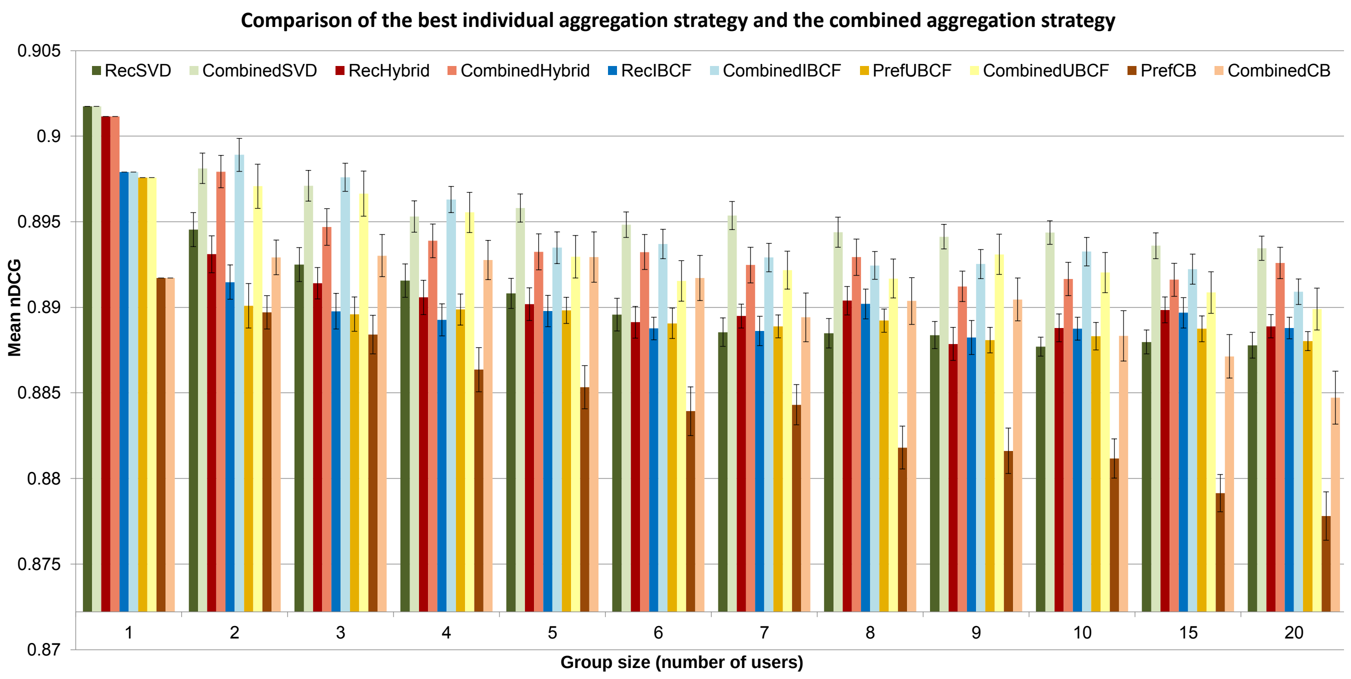


Figure 1: The accuracy of the group recommendations calculated using the best individual aggregation strategy and the combined aggregation strategy

Table 1: Statistical T-test comparing the best individual aggregation strategy and the combined aggregation strategy for groups with size=5

Algorithm	t(58)	p-value
SVD	-4.39	**0.00**
Hybrid	-2.53	**0.01**
IBCF	-2.33	**0.02**
UBCF	-2.66	**0.01**
CB	-3.55	**0.00**

tions. Both existing data aggregation strategies are combined to make a more informed decision hereby reducing the risk of recommending undesirable or less suitable items to the group. The results show that the combination of aggregation strategies outperforms the individual aggregation strategies for various sizes of the group and in combination with various recommendation algorithms. The proposed aggregation strategy can be used to increase the accuracy of (commercial) group recommender systems.

7. REFERENCES

[1] L. Baltrunas, T. Makcinskas, and F. Ricci. Group recommendations with rank aggregation and collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 119–126, New York, NY, USA, 2010. ACM.

[2] S. Berkovsky and J. Freyne. Group-based recipe recommendations: analysis of data aggregation strategies. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 111–118, New York, NY, USA, 2010. ACM.

[3] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, UAI'98, pages 43–52, San Francisco, CA, USA, 1998.

[4] Y.-L. Chen, L.-C. Cheng, and C.-N. Chuang. A group recommendation system with consideration of interactions among group members. *Expert Systems with Applications*, 34(3):2082 – 2090, 2008.

[5] T. De Pessemier, S. Dooms, and L. Martens. Design and evaluation of a group recommender system. In *Proceedings of the sixth ACM conference on Recommender systems*, RecSys '12, pages 225–228, New York, NY, USA, 2012. ACM.

[6] S. Dooms, T. De Pessemier, and L. Martens. A user-centric evaluation of recommender algorithms for an event recommendation system. In *Proceedings of the workshop on User-Centric Evaluation of Recommender Systems and Their Interfaces at ACM Conference on Recommender Systems (RECSYS)*, pages 67–73, 2011.

[7] J. Masthoff. Group modeling: Selecting a sequence of television items to suit a group of viewers. *User Modeling and User-Adapted Interaction*, 14:37–85, 2004.

[8] L. Quijano-Sanchez, J. A. Recio-Garcia, and B. Diaz-Agudo. Personality and social trust in group recommendations. In *Proceedings of the 2010 22nd IEEE International Conference on Tools with Artificial Intelligence - Volume 02*, ICTAI '10, pages 121–126, Washington, DC, USA, 2010. IEEE Computer Society.

[9] Telematica Instituut / Novay. Duine Framework, 2009. Available at `http://duineframework.org/`.

[10] The Apache Software Foundation. Apache Mahout, 2012. Available at `http://mahout.apache.org/`.

Simplifying Privacy Decisions: Towards Interactive and Adaptive Solutions

Bart P. Knijnenburg
Donald Bren School of Information and Computer Sciences
University of California, Irvine
bart.k@uci.edu

1. INTRODUCTION

Privacy concerns are an important barrier to the growth of social networks, e-commerce, ubiquitous computing, and location sharing services. The large majority of Internet users takes a pragmatic stance on information disclosure: they trade off the anticipated benefits with the risks of disclosure, a decision process that has been dubbed *privacy calculus* [10,23]. Privacy decisions are inherently difficult though, because they have delayed and uncertain repercussions that are difficult to trade-off with the possible immediate gratification of disclosure [3,5].

How can we help users to balance the benefits and risks of information disclosure in a user-friendly manner, so that they can make good privacy decisions? Existing research has explored two approaches to this problem, but neither provides a satisfying solution. Below I discuss these two approaches, and introduce a new *user-tailored* approach that provides more user-friendly privacy decision support.

2. TRANSPARENCY AND CONTROL

To help users with their privacy calculus, experts recommend giving users comprehensive *control* over what data they wish to share, and more *transparency* about the implications of their decisions [1,22]. However, while users claim to *want* full control over their data, they avoid the hassle of actually *exploiting* this control [8]. Moreover, the privacy controls of systems like Facebook are so complex that users do not even seem to know the implications of their own settings [25]. Similarly, informing users about the rationale behind information requests does not make them more discerning about their privacy decisions, but merely makes them worry about privacy in general. For example, displaying a privacy label on an e-commerce website—a supposed vote of confidence—may *decrease* instead of increase purchases [7].

Evidently, transparency and control do not work well in practice. Due to the complexity of privacy decisions and users' bounded rationality [2,3], an increase in transparency and control often just aggravates the problem by introducing choice overload [12,27] and information overload [11].

3. PRIVACY NUDGES

An alternative approach to support privacy decisions is to introduce subtle yet persuasive *nudges*. Carefully designed nudges make it easier for people to make the right choice, without limiting their ability to choose freely [29]. A *justification*, for example, makes it easier to rationalize decisions and to minimize the regret associated with choosing the wrong option [9]. The effect of justifications in privacy research seems to vary. In my own research I have found that justifications are regarded as helpful, but do not increase users' disclosure or satisfaction but rather *decrease* them [18,19]. Sensible *defaults* are another type of nudge that strongly impact disclosure [4,14,19]. Examples are framing a disclosure decision as either opt-in or opt-in, or changing the order of information requests.

The problem with nudges is that they take a one-size-fits-all approach to privacy: They assume that the "true cost" [13] of disclosure is roughly the same for every user, piece of information, and situation. But privacy decisions are highly user- and context-dependent: The fact that one person has no problems disclosing a certain item in a particular context does not mean that disclosure is equally likely for a different person, a different item, or in a different context [16,24]. Likewise, what is a convincing justification to disclose a certain item in a particular context for a certain person, may be a completely irrelevant reason for a different person, a different item, or a different context [6,21]. What we need, then, is *personalized* privacy decision support.

4. EXPLICATING PRIVACY

The first step towards personalized privacy decision support is to *explicate* the privacy calculus: to move beyond a mere description towards deeper understanding of people's cognitive decision-making process. What kind of benefits and threats do users consider when making disclosure decisions? What is the relative weight of each of these aspects? Can the weights be influenced by a justification or a default, and if so, in what context(s)? More research is needed to answer these questions.

For example, I showed in [19] that the effect of justifications on information disclosure decisions is mediated by users' perceptions of help, trust and self-anticipated satisfaction with the system. In [17], I demonstrated that the effect of decision context (i.e. the available options) in a location-sharing service depends on users' perception of the privacy and benefits of the available options. Finally, in [20] we show that perceived risk and perceived relevance mediate users' evaluation of the purpose-specificity of information disclosure requests.

5. CONTEXTUALIZING PRIVACY

The second step towards a personalized privacy decision support is to *contextualize* the privacy calculus: to determine how stable information disclosure is across people, items and situations, and, importantly, where it is context-dependent.

For example, my research shows that although justifications generally do not increase disclosure or satisfaction, tailoring justifications to the user can reduce this negative effect [21]. Such tailored justifications are *personalized privacy nudges*: they intelligently choose the correct justification for the respective user, or decide to not show any justification at all.

Similarly, *personalized defaults* can be set up in a way that anticipates people's disclosure behavior, thereby making the disclosure decisions easier and more convenient. My work and that of others shows that even though privacy preferences vary considerably across users, distinct subgroups of users with similar privacy preferences can be identified in many domains [16,26]. Moreover, these subgroups can be mapped to demographics (e.g. age) and other behaviors (e.g. mobile Internet usage). My recent work shows that these personalized defaults may also be tailored to the website requesting the information: in [20] we show that people are more likely to disclose information that matches the *purpose* of the website requesting the information.

Finally, in [17] I show that privacy decisions are influenced by the *available options* to choose from ("context effects", cf. [15,28,30]). In that study, users of a location-sharing service decided whether to share their location with friends, colleagues and third party applications, with the following options: no sharing, city, city block, or exact location. We manipulated the availability of the "city" and "exact location" options, and showed that their absence or presence had a strong impact on how many users would choose each of the other available options.

6. PRIVACY ADAPTATION PROCEDURE

The ultimate purpose of this contextualized and explicated understanding of users' privacy calculus is to develop a Privacy Adaptation Procedure to support people's privacy decisions. Using recommender system algorithms, the procedure predicts users' privacy preferences based on their known characteristics. It then provides automatic "smart default" settings in line with users' disclosure profile. Smart defaults reduce the burden of control, but at the same time respect users' inherent privacy preferences. Similarly, it provides tailored disclosure justifications, but only to users who can be expected to react rationally to them, so that they will not cause privacy scares in the other users.

This Privacy Adaptation Procedure relieves some of the burden of the privacy decision from the user by providing the right amount of information and control that is useful but not overwhelming or misleading. It thus enables users to make privacy-related decisions within the limits of their bounded rationality.

7. REFERENCES

1. Acquisti, A. and Gross, R. Imagined Communities: Awareness, Information Sharing, and Privacy on the Facebook. In *Privacy Enhancing Technologies*. 2006, 36–58.
2. Acquisti, A. and Grossklags, J. Privacy and Rationality in Individual Decision Making. *IEEE Security & Privacy 3*, 1 (2005), 26–33.
3. Acquisti, A. and Grossklags, J. What Can Behavioral Economics Teach Us About Privacy? In A. Acquisti et al., eds., *Digital Privacy*. Taylor & Francis, 2008, 363–377.
4. Acquisti, A., John, L.K., and Loewenstein, G. The Impact of Relative Standards on the Propensity to Disclose. *Journal of Marketing Research 49*, 2 (2012), 160–174.
5. Acquisti, A. Privacy in Electronic Commerce and the Economics of Immediate Gratification. *Proceedings of the ACM Conference on Electronic Commerce*, (2004), 21–29.
6. Besmer, A., Watson, J., and Lipford, H.R. The impact of social navigation on privacy policy configuration. *Proceedings of SOUPS*, (2010).
7. Bustos, L. Best Practice Gone Bad: 4 Shocking A/B Tests. *GetElastic*, 2012. http://www.getelastic.com/best-practice-gone-bad-4-shocking-ab-tests/.
8. Compañó, R. and Lusoli, W. The Policy Maker's Anguish: Regulating Personal Data Behavior Between Paradoxes and Dilemmas. In T. Moore, D. Pym and C. Ioannidis, eds., *Economics of Information Security and Privacy*. Springer US, New York, NY, 2010, 169–185.
9. Connolly, T. and Zeelenberg, M. Regret in decision making. *Current directions in psych. science 11*, 6 (2002), 212–216.
10. Culnan, M.J. "How Did They Get My Name?": An Exploratory Investigation of Consumer Attitudes toward Secondary Information Use. *MISQ 17*, 3 (1993), 341–363.
11. Eppler, M.J. and Mengis, J. The Concept of Information Overload: A Review of Literature from Organization Science, Accounting, Marketing, MIS, and Related Disciplines. *The Information Society 20*, 5 (2004), 325–344.
12. Iyengar, S.S. and Lepper, M.R. When choice is demotivating: Can one desire too much of a good thing? *J. of Personality and Social Psychology 79*, 6 (2000), 995–1006.
13. John, L.K., Acquisti, A., and Loewenstein, G. Strangers on a Plane: Context-Dependent Willingness to Divulge Sensitive Information. *J of Consumer Research 37*, 5 (2011), 858–873.
14. Johnson, E.J., Bellman, S., and Lohse, G.L. Defaults, Framing and Privacy: Why Opting In ≠ Opting Out. *Marketing Letters 13*, 1 (2002), 5–15.
15. Kahneman, D. and Tversky, A. Prospect Theory: An Analysis of Decision under Risk. *Econometrica 47*, 2 (1979), 263–292.
16. Knijnenburg, B.P., Kobsa, A., and Jin, H. Dimensionality of information disclosure behavior. *International Journal of Human-Computer Studies*, (2013).
17. Knijnenburg, B.P., Kobsa, A., and Jin, H. Preference-based location sharing: are more privacy options really better? *Proceedings of CHI*, (2013), 2667–2676.
18. Knijnenburg, B.P., Kobsa, A., and Saldamli, G. Privacy in Mobile Personalized Systems: The Effect of Disclosure Justifications. *Proceedings of U-PriSM*, (2012).
19. Knijnenburg, B.P. and Kobsa, A. Making Decisions about Privacy: Information Disclosure in Context-Aware Recommender Systems. *ACM Transactions on Interactive Intelligent Systems 3*, 3 (2013).
20. Knijnenburg, B.P. and Kobsa, A. Counteracting the negative effect of form auto-completion on the privacy calculus. *Proceedings of ICIS*, (2013).
21. Knijnenburg, B.P. and Kobsa, A. Helping users with information disclosure decisions: potential for adaptation. *Proceedings of IUI*, (2013), 407–416.
22. Kolter, J. and Pernul, G. Generating User-Understandable Privacy Preferences. *2009 International Conference on Availability, Reliability and Security*, IEEE Computer Society (2009), 299–306.
23. Laufer, R.S. and Wolfe, M. Privacy as a Concept and a Social Issue: A Multidimensional Developmental Theory. *Journal of Social Issues 33*, 3 (1977), 22–42.
24. Li, H., Sarathy, R., and Xu, H. Understanding situational online information disclosure as a privacy calculus. *Journal of Computer Information Systems 51*, 1 (2010), 62–71.
25. Liu, Y., Gummadi, K.P., Krishnamurthy, B., and Mislove, A. Analyzing facebook privacy settings: user expectations vs. reality. *Proc. SIGCOMM 2011*, ACM (2011), 61–70.
26. Phelps, J., Nowak, G., and Ferrell, E. Privacy Concerns and Consumer Willingness to Provide Personal Information. *Journal of Public Policy & Marketing 19*, 1 (2000), 27–41.
27. Scheibehenne, B., Greifeneder, R., and Todd, P.M. Can There Ever Be Too Many Options? A Meta-Analytic Review of Choice Overload. *Journal of Consumer Research 37*, 3 (2010), 409–425.
28. Simonson, I. Choice Based on Reasons: The Case of Attraction and Compromise Effects. *Journal of Consumer Research 16*, 2 (1989), 158–174.
29. Thaler, R.H. and Sunstein, C. *Nudge : improving decisions about health, wealth, and happiness*. Yale University Press, New Haven, NJ & London, U.K., 2008.
30. Tversky, A. Elimination by aspects: A theory of choice. *Psychological Review 79*, 4 (1972), 281–299.

Food Recommendations: Biases that Underpin Ratings

Shlomo Berkovsky, Jill Freyne
CSIRO Computational Informatics
Marsfield, NSW, Australia
firstname.lastname@csiro.au

ABSTRACT
The context in which preference information is gathered from users is recognised as one of the important factors in generating user models. Much research has been carried out into sensing and recording the physical and environmental contexts of user interactions, and it was shown that these affect the derived user modelling data. In this talk we introduce a new facet of context called a *Reasoning Context*, which could be informed by intrinsic factors, such as emotions, mood, biases, or user's thought patterns. If deciphered from the observable rating patterns, this contextual facet can be leveraged both in the user modelling and recommendation stages of the personalisation process.

We illustrate the discovery, stability, and exploitation of the Reasoning Context through a case study that uses a dataset of user ratings for a set of recipes. Each recipe was associated with a pre-populated set of domain features: the cuisine type, the main ingredient, the number of ingredients required, and the cooking complexity of the recipe. We applied a feature selection algorithm to uncover user reasoning patterns in the gathered rating data. The analysis showed which features each user appears to be reasoning on when providing their recipe ratings. For instance, some users were found to reason only on the main ingredient of a recipe, while others appeared to apply more complex reasoning that considered multiple features. We showed that the observed reasoning patterns and bias were stable and predictive.

The value of uncovering the users' reasoning patterns and using these as part of the Reasoning Context is two-fold. By knowing what features are important to users, a user modelling system could adjust the rating acquisition process and obtain ratings bearing high-value information, thus, reducing the data gathering load and alleviating the cold start problem. This knowledge can be invaluable when delivering recommendations aimed at diversity, serendipity and content discovery, since it can fuel an individual user-tailored diversity metric and lead to better recommendations.