P R O C E E D I N G S

The 7th International Workshop on

Intelligent Techniques for Web Personalization & Recommender Systems

ITWP 2009

Editors:

Sarabjot Singh Anand, Bamshad Mobasher, Alfred Kobsa, Dietmar Jannach

July 11, 2009
Pasadena, California, USA

In conjunction with

The 21st International Joint Conference on Artificial Intelligence - IJCAI 2009

Workshop Co-Chairs

Sarabjot Singh Anand, University of Warwick, UK
Bamshad Mobasher, DePaul University, Chicago, USA
Alfred Kobsa, University of California, Irvine, USA
Dietmar Jannach, Technische Universität Dortmund, Germany

Program Committee

Esma Aimeur, Université de Montréal, Canada
Gediminas Adomavicius, CSOM, University of Minnesota
Liliana Ardissono, University of Torino, Italy
Bettina Berendt, K.U.Leuven, Belgium
Shlomo Berkovsky, University of Melbourne, Australia
José Luís Borges, University of Porto, Portugal
Derek Bridge, University College Cork, Ireland
Robin Burke, DePaul University
Alexander Felfernig, Graz University of Technology, Austria
Gerhard Friedrich, University Klagenfurt, Austria
Rayid Ghani, Accenture, USA
Marko Grobelnik, Jožef Stefan Institute, Slovenia
Andreas Hotho, University of Würzburg, Germany
Alípio Jorge, University of Porto, Portugal
Mark Levene, University College, London, UK
Stuart E. Middleton, University of Southampton, UK
Alexandros Nanopulos, Aristotle University of Thessaloniki, Greece
Olfa Nasraoui, University of Louisville
Claire Nedellec, Université Paris Sud, Paris, France
Seung-Taek Park, Yahoo Inc.
George Paliouras, Demokritos National Centre for Scientific Research, Greece
David Pennock, NEC Research Institute, USA
Naren Ramakrishnan, Virginia Tech
Francesco Ricci, Free University of Bozen-Bolzano, Italy
Lars Schmidt-Thieme, University of Hildesheim, Germany
Spiros Sirmakessis, University of Patras, Greece
Barry Smyth, University College Dublin, Ireland
Markus G. Stolze, IBM Watson Research Center, NY
Suk-Chung Yoon, Widener University, Pennsylvania
Markus Zanker, University Klagenfurt, Austria
Daniel Zeng, University of Arizona

Foreword

Web Personalization can be defined as any set of actions that can tailor the Web experience to a particular user or set of users. The experience can be something as casual as browsing a Web site or as (economically) significant as trading stocks or purchasing a car. The actions can range from simply making the presentation more pleasing to anticipating the needs of a user and providing customized and relevant information. To achieve effective personalization, organizations must rely on all available data, including the usage and clickstream data (reflecting user behavior), the site content, the site structure, domain knowledge, as well as user demographics and profiles. Efficient and intelligent techniques are needed to mine this data for actionable knowledge, and to effectively use the discovered knowledge to enhance the users' Web experience. These techniques must address important challenges emanating from the size of the data, the fact that they are heterogeneous and very personal in nature, as well as the dynamic nature of user interactions with the Web. These challenges include the scalability of the personalization solutions, data integration, and successful integration of techniques from machine learning, information retrieval and filtering, databases, agent architectures, knowledge representation, data mining, text mining, statistics, information security and privacy, user modeling and human-computer interaction.

Recommender systems represent one special and prominent class of such personalized Web applications, which particularly focus on the user-dependent filtering and selection of relevant information and – in an e-Commerce context - aim to support online users in the decision-making and buying process. Recommender Systems have been a subject of extensive research in AI over the last decade, but with today's increasing number of e-commerce environments on the Web, the demand for new approaches to intelligent product recommendation is higher than ever. There are more online users, more online channels, more vendors, more products and, most importantly, increasingly complex products and services. These recent developments in the area of recommender systems generated new demands, in particular with respect to interactivity, adaptivity, and user preference elicitation. These challenges, however, are also in the focus of general Web Personalization research.

In the face of this increasing overlap of the two research areas, the aim of this workshop is to bring together researchers and practitioners of both fields, to foster an exchange of information and ideas, and to facilitate a discussion of current and emerging topics related to "Web Intelligence", particularly regarding its application in recommender systems. This workshop represented the seventh in a successful series of ITWP workshops that have been held at IJCAI and AAAI.

This year’s workshop attracted a number of high-quality contributions from 15 different countries. Of these, 7 papers (less than 40%) were accepted for full presentation at the workshop, with an additional 3 accepted for short presentations. The accepted papers deal with a wide variety of issues and techniques for creating more intelligent personalization systems, but generally fell into a four broad categories: Modeling and Personalization Strategies; Recommendation Algorithms; Hybrid Recommenders and Enabling Technologies for Recommendation. The workshop also features an invited talk by Barry Smyth, University College Dublin on “Personalization and Collaboration in Social Search”

ITWP 2009 Organizing Committee

July 2009, Pasadena, USA

Table of Contents

Long Papers

Short Papers

ITWP 2009 Workshop Program (July 11, 2009)

08:30-08:45 Opening

Technical session 1 - Modeling and Personalization Strategies

08:45-09:10 Online Selection of Mediated and Domain-Specific Predictions for Improved Recommender Systems
Stephanie Rosenthal, Manuela Veloso, Anind Dey

09:10-09:35 Using Gaussian Spatial Processes to Model and Predict Interests in Museum Exhibits
Fabian Bohnert, Ingrid Zukerman, and Daniel F. Schmidt

09:35-10:00 Optimal Set Recommendations based on Regret
Paolo Viappiani, Craig Boutilier

Coffee break (10:00 - 10:30)

Technical session 2 – Enabling Technologies

10:30-10:55 Analysis of Web Usage Patterns in Consideration of Various Contextual Factors
Jinhyuk Choi, Jeongseok Seo and Geehyuk Lee

10:55-11:20 Uncovering Functional Dependencies in MDD-Compiled Product Catalogues
Tarik Hadzic and Barry O'Sullivan

11:20-11:35 Intelligent Web Navigation Using Virtual Assistants
Eduardo Eisman

Lunch (11:35 - 13:15)

***Invited Talk* Barry Smyth, University College Dublin**

13:15-14:10 Personalization and Collaboration in Social Search

Technical session 3 – Recommendation Algorithms

14.10-14:35 Effectiveness of different recommender algorithms in the Mobile Internet: A case study
Kolja Hegelich and Dietmar Jannach

14:35-15:00 Adapting K-Nearest Neighbor for Tag Recommendation in Folksonomies
Jonathan Gemmell, Thomas Schimoler, Maryam Ramezani, Bamshad Mobasher

Coffee break (15:00 - 15:30)

Technical session 4 – Hybrid Recommenders

15:30-15:45 Collaborative Filtering With Adaptive Information Sources
Neal Lathia, Xavier Amatriain and Josep M. Pujol

15:45–16:00 Exploiting Semantic Web Technologies for Recommender Systems: A Multi View Recommendation Engine
Houda Oufaida

16.00-17:00 Wrap-up and discussion

Online Selection of Mediated and Domain-Specific Predictions for Improved Recommender Systems

Stephanie Rosenthal, Manuela Veloso, Anind Dey
School of Computer Science
Carnegie Mellon University
{srosenth,veloso,anind}@cs.cmu.edu

Abstract

Recommender systems use a set of reviewers and advice givers with the goal of providing accurate user-dependent product predictions. In general, these systems assign weights to different reviewers as a function of their similarity to each user. As products are known to be from different domains, a recommender system also considers product domain information in its predictions. As there are few reviews compared to the number of products, it is often hard to set the similarity-based weights as there is not a large enough subset of reviewers who reviewed the same products. It has then been recently suggested that not considering domains will increase the amount of reviewer data and the overall prediction accuracy in a *mediated* way. However, clearly, if different reviewers are similar to a user in each product domain, then *domain-specific* predictions could be superior to mediated ones.

In this paper, we consider two advice giver algorithms to provide *domain-specific* and *mediated* predictions. We analyze both advice giver algorithms using large real data sets to characterize when each is more accurate for users. We realize that for a considerable number of users, the *domain-specific* predictions are possible and more accurate. We then contribute an improved general recommender system algorithm that autonomously selects the most accurate mediated or domain-specific advice giver *for each user*. We validate our analysis and algorithm using real data sets and show the improved predictions for different users.

Introduction

We model a product recommender system as a set of reviews defined by reviewers, product domains (*e.g.,* DVDs, books, clothes), and advice givers. Users request that the recommender system provide predictions of whether they will like a set of products of their choosing. Users then have the option of providing their own reviews of those products. As the advice giver that makes product predictions receives the user's actual reviews, it assigns *domain-specific* weights to the reviewers as a function of the similarity between their reviews and the user's. The reviewers whose reviews are most similar to the user's receive higher weight. The advice giver uses the weights of the reviewers and their reviews to guide its predictions with the goal of providing the most accurate predictions.

Because reviewers review a relatively small number of products, it is difficult to find enough reviewers with similar reviews to make accurate predictions for every product each user requests. The problem is exacerbated as products are divided into domains so there are fewer product reviews to train the domain-specific weights with. Resolving the data sparsity problem has been the focus of much recommender system work. Although it is widely accepted that *domain-specific* reviewers result in accurate predictions, it has recently been suggested that a *mediated* advice giver that combines multiple domains of products and holds only a single set of weights for each, user would help alleviate the data sparsity problem (Berkovsky, Kuflik, and Ricci 2007a).

While learning only one set of weights will increase the amount of data to train with, there is an underlying assumption that the reviewers with similar reviews to a user in one product domain (*e.g.,* DVDs) will also have similar reviews to that user in other domains (*e.g.,* books and clothes). While a domain-specific advice giver captures these differences, the mediated advice giver does not. Intuitively, it seems unlikely that for all users there is a set of reviewers with similar reviews in all domains of products. The focus of this work is to understand in real recommender data sets how data sparsity and the user's reviews affect the weights of the reviewers and the accuracy of the advice givers' predictions.

We first present an overview of how advice givers weigh reviewers and make predictions for users and give examples of weights that affect the two advice givers' accuracy. We show using data from two large recommender systems that potentially half of the users benefitted from the mediated advice giver while the other half required domain-specific weights. Additionally we find in a third and more sparse recommender data set that both advice givers have equal accuracies when reviewers do not provide reviews for more than one category. We, then, show how accuracy changes for each advice giver as reviewers review products in more categories. Assuming that reviewers do provide reviews in more categories (as found in the first two data sets) and because different users require the two advice givers equally, we contribute two online user-dependent selection algorithms for the recommender system to choose which advice giver

makes the highest accuracy predictions for each user. Finally, we validate both our initial findings and the selection algorithms with a fourth recommender system data set and conclude that each user benefits from the user-dependent selection rather than a recommender system that uses one type of advice giver.

Advice Givers

A recommender system is comprised of a set of products with corresponding domain information, reviews R and an advice giver. The set of reviews R is an $M \times N$ matrix of M reviewers and N products. The review R_{ij} is a discrete value $v \in V$ that reviewer r_i provides for product p_j. Possible values V may be binary $\{0, 1\}$ or ranging over a subset of the integers (*e.g.,* $\{1, 2, 3, 4, 5\}$). Each product p_j is assigned a domain $d \in D$.

Domain-Specific Advice Giver

An advice giver's task is to provide personalized predictions $v \in V$ of products p that a user u requests. In order to provide personalized predictions for each user, the *Domain-Specific Advice Giver (DSAG)* assigns a weight $w_i^{u,d}$ to each reviewer for each user and domain d (See Algorithm 1). The weight of a reviewer is related to how often the reviewer gave review values similar to the user's reviews and are modeled after experts algorithms ((Auer et al. 1995; Littlestone and Warmuth 1994)) which have been used widely in predicting reviews (*e.g.,* (Nakamura and Abe 1998)). These weights are initially uniform across the reviewers for each user (Line 1)

$$\forall u, d, i \; w_i^{u,d} = 1/M \tag{1}$$

The reviewers are the "experts," and the advice giver makes a prediction by polling the reviewers as a function of the weights assigned to them. The advice giver uses weighted majority to make a prediction for a product in domain d_k that a user requests, by summing the weights of reviewers that provide each value v, and predicts the value with the most weight:

$$argmax_v \sum_i I(R_{ij} == v) * w_i^{u,k} \tag{2}$$

where I is the identity function that returns $I(true) = 1$ and $I(false) = 0$ (Lines 2-4) ((Littlestone and Warmuth 1994)). The advice giver updates the weights as a function of the distance between the reviewer's reviews and the user u's later actual review a_j for the product p_j (Lines 5,6):

$$w_i^{u,k} = \frac{exp(ln(w_i^{u,k}) - \ell_1(R_{ij}, a_j))}{\sum_h exp(ln(w_h^{u,k}) - \ell_1(R_{hj}, a_j))} \tag{3}$$

Because the advice giver makes predictions about the user's review but does not know the actual review ahead of time, the weights for the domain d_k are recalculated online as the user provides reviews for products in that domain. We implement the sleeping experts algorithm to reweigh only the reviewers that provided a review for the product (Freund et al. 1997; Blum and Mansour 2005). If a reviewer does not provide a review for the product p_j, its weight does not change. The goal of the advice giver is to weigh the reviewers for each user such that the resulting predictions are as accurate as possible compared to the hindsight knowledge of the users' reviews.

Algorithm 1 Domain-Specific Advice Giver (DSAG)

1: For a new user u, initialize $w_i^{u,c}$ according to (1)
2: **for all** products p_j **do**
3: $k \leftarrow$ domain(p_j)
4: Predict according to (2)
5: **if** user u gives review a_j **then**
6: Update $w_i^{u,k}$ according to (3)
7: **end if**
8: **end for**

Mediated Advice Giver

The DSAG can provide precise predictions for users in each domain, but it requires enough reviews in R to cover all products with enough reviewers and requires the user provide enough reviews in each domain to reweigh the reviewers enough times for the weights to converge. In typical recommender systems, however, the review matrix R is very sparse in both the number of reviews provided for a particular product and the number of reviews provided by a particular reviewer. Because of this sparsity, the number of reviewers that get reweighed for any given product that the user requests is far fewer than the total number of reviewers. As a result, the DSAG requires the user to review a lot of products before it can provide accurate enough predictions. This problem is exacerbated because the products are often split into domains and the algorithms require the user to review the same number of products in each domain to predict accurately in each. The focus of much recommender system research has centered around resolving this data sparsity problem (Adomavicius and Tuzhilin 2005).

While most work has focused on *hybrid* recommender systems to increase accuracy by combining different weighing techniques (*e.g.,* (Burke 2002; Umyarov and Tuzhilin 2007)), one recent idea is to combine domains to increase the number of products that affect the reviewers' weights. One idea is to keep domain-specific weights, but to allow the DSAG to reference all of a reviewer's weights to determine if that reviewer is similar to the user in any domain (Berkovsky, Kuflik, and Ricci 2007b). If there is not enough information about a particular domain to make sug-

Algorithm 2 Mediated Advice Giver (MAG)

1: For a new user u, initialize $\forall i \; w_i^u \leftarrow \frac{1}{|M|}$
2: **for all** products p_j **do**
3: Predict $v = argmax_v \sum_i I(R_{ij} == v) * w_i^u$
4: **if** user reports review a_j **then**
5: $w_i^u \leftarrow \frac{exp(ln(w_i^u) - \ell_1(R_{ij}, a_j))}{\sum_h exp(ln(w_h^u) - \ell_1(R_{hj}, a_j))}$
6: **end if**
7: **end for**

(a)

R	p_1 d_1	p_2 d_2	p_3 d_1	p_4 d_2
r_1	5	2	5	3
r_2	3	5	2	5
r_3	3	2	2	3
u	5	5	5	5

(b)

$w^{u,c}$	t_0	t_1	t_2	t_3	t_4
$w_1^{u,1}$	.33	.78	.78	.99	.99
$w_2^{u,1}$	.33	.11	.11	.005	.005
$w_3^{u,1}$	.33	.11	.11	.005	.005
$w_1^{u,2}$	.33	.33	.045	.045	.005
$w_2^{u,2}$	.33	.33	.91	.91	.99
$w_3^{u,2}$	.33	.33	.045	.045	.005

(c)

w^u	t_0	t_1	t_2	t_3	t_4
w_1^u	.33	.78	.25	.87	.498
w_2^u	.33	.11	.71	.12	.498
w_3^u	.33	.11	.04	.01	.004

Table 1: Example. (a) The review matrix R contains 4 products and 3 reviewers. (b) The DSAG recalculates domain-specific weights as the user provides their actual reviews to find that r_1 and r_2 are most similar for domains d_1 and d_2, respectively. (c) The MAG recalculates the single set of weights as the user provides reviews and finds r_1 and r_2 to be equally similar.

gestions about a product, the system could take advantage of the user's similar reviewers in other domains to make predictions. *Mediation*, on the other hand, combines the weights from multiple domains together (Berkovsky, Kuflik, and Ricci 2007a).

A Mediated Advice Giver (MAG) makes domain-independent predictions with the expectation that the advice giver will identify the most similar reviewers sooner, and provide more accurate predictions with sparse matrices, because all products affect the same weights. The Mediated Advice Giver (Algorithm 2) is the same as the DSAG, except that a single set of weights is maintained which is updated for every product in every domain. The MAG is accurate when there is a consistent set of similar reviewers to a user for every domain (reviewers have proportional weights in all domains). However, when different reviewers have high weights in different domains, the MAG weighs all equally, which can result in poor predictions. Intuitively, if a user had reviews similar to one set of reviewers about DVDs and very different reviews about books, the DSAG, which holds different weights for DVDs and books, would provide more accurate predictions.

Example Suppose a new user joins a recommender system that uses a Domain-Specific Advice Giver with three reviewers ($M = 3$) r_1, r_2, r_3, that review four products ($N = 4$) p_1, p_2, p_3, p_4 with reviews presented in Table 1(a). The values are shown for later use in the example. $D = \{d_1, d_2\}$ are assigned to the products and the possible values that the reviewer can give and advice giver can predict are $V = \{1, 2, 3, 4, 5\}$. The DSAG initializes the weights $w_i^{u,d}$ to $1/M = 1/3$ for both domains (Table 1(b) column t_0). The user requests a prediction for product p_1 in domain d_1. The advice giver calculates which value to predict using the initial weights and chooses to predict 3 because it has the maximum weight associated with it. The user notifies the advice giver that their actual review r_1 of p_1 is 5 (last row of Table 1(a)) and the advice giver uses that information to reweigh the reviewers for domain d_1 (Table 1(b) column t_1). Then the user requests a prediction for p_2, and the advice giver uses the new initialized weights for domain d_2 to predict 2. The user responds with value 5 and the advice giver recalculates the weights from domain d_2 (column t_2). This continues for all 4 products.

The weights (shown in Table 1(b)) do not change on the time steps when the advice giver predicts a value for a product that is from a different domain. At the end, we can see that the user has the same reviews as r_1 for domain d_1 and the same reviews for r_2 as d_2 and the DSAG correctly identifies them as most similar by assigning them highest weight.

Now suppose that R is the same, but the recommender system uses a MAG to predict for the user. Because reviewers 1 and 2 are each correct 50% of the time, their weights change over the four products (See Table 1(c)). The MAG assigns higher weight to the wrong reviewer and predicts 2 or 3 when it should predict 5 for all products.

Approach

The focus of this work is to determine whether data sparsity consistently affects the accuracy of the DSAG for all users and choose the most accurate advice giver for each user. We will first show, using synthetic data, different weight distributions for users in the MAG and DSAG and analyze their prediction accuracies. We then show, using three real recommender system data sets, that both the MAG and DSAG give more accurate predictions for some users. Additionally, for very sparse data sets, we find that both advice givers can give identical results. We analyze this phenomenon and provide accuracy results on synthetic data sets with these properties. Because neither advice giver can be excluded as less accurate, we provide two algorithms to dynamically select which advice giver to use for each user and validate our results and algorithms on a fourth recommender data set.

Advice Giver Evaluation - Synthetic Data

In this section, we evaluate the worst-case and more realistic review matrices and user preferences to better understand the performance of the MAG and DSAG under different amounts of data sparsity and different reviewers. We show that the MAG converges faster on the weight distribution, assuming those best reviewers are the same across categories. Then, we will show a more extreme case of the example above where a user agrees strongly with one reviewer in each category and disagrees strongly with the rest, causing the DSAG to perform better than the MAG. When we relax the constraint of a different "best" reviewer in each category, the MAG and DSAG perform equally well. Finally, we relax the assumption that all reviewers review all products and explore very sparse review matrices. We show

that in cases where reviewers only review products in a particular category, the two advice givers perform equally well. We will use these results later to analyze our results from three real recommender system data sets.

Sparse Data

For the following examples, we will assume that users have polar preferences - either strongly disliking (1) or strongly liking (5) each product. For simplicity, the user will always strongly like the product and give it a 5. Also for simplicity, we will only have m reviewers, m categories, and $n >> m$ products that are evenly distributed across the categories.

The MAG can converge on a single weight distribution using all of the products while the DSAG instead uses m weight distributions - one for each category. Assuming that each of the m categorical weight distributions are similar, the DSAG will converge on each distribution separately although it turns out they should all be the same. If each weight distribution takes the same amount of time to converge, the DSAG will take longer to come to the same conclusion as the MAG. If the products are not distributed evenly across categories, it could take the DSAG much longer to converge on the rare categories. As an example, we define the review matrix in the following way:

$$R_{ij} = \begin{cases} 5 & (p \wedge i = 1) \\ 1 & ((1-p) \wedge i = 1) \\ 1 & i \neq 1 \end{cases}$$

For all categories, reviewer 1 is correct p percent of the time and gives the same review as the rest of the reviewers (1-p) percent of the time. The DSAG will have to find this pattern for each weight distribution while the MAG only finds it once. Because this is a simple distribution and all reviewers give reviews for all products, it takes relatively few products to find the pattern. The MAG finds the right weight distribution after $1/p$ products while the DSAG takes m/p products to converge all distributions.

In general, it takes $1/p$ product reviews to find the weight of each reviewer for each weight distribution. If not all reviewers provide reviews for each product, it could take much longer to converge. The MAG also assumes that the weight distributions for each category are similar. If they are not, combining them together into the single distribution may cause prediction errors.

Categorical Weight Distributions

The DSAG can perform well compared to the MAG when the reviewers have very different weights in each category as shown in the example above. As a more extreme example, we define the review matrix in the following way:

$$R_{ij} = \begin{cases} 5 & \text{product j in category i} \\ 1 & \text{product j not in category i} \end{cases}$$

It is quite obvious to see that if the user always says 5, that reviewer i always has the highest weight in category i and the rest of the reviewers have almost 0 weight. The weight distribution for each category is very different. The DSAG converges on the correct weights very quickly because of the high degree of similarity between reviewers and the user. The MAG, alternatively, sees an equal number of products in each category and converges on a uniform weight distribution across reviewers. Because more reviewers recommend the value 1 for each product, the MAG predicts incorrectly every time. We tested this hypothesis with data generated with the above rule on recommendation systems with the weighted majority algorithm. The error rates of the systems were calculated. The similarity distributions were also examined at the end of the trials to compare to the expected distribution.

Advice Giver	Accuracy
DSAG	100%
MAG	0%

Table 2: The domain-specific advice giver is 100% accurate while the mediated advice giver is 0% accurate.

Table 2 shows the prediction accuracy by advice giver. As expected, there is a significant drop in accuracy when combining the domain-specific advice givers. The MAG converges on a uniform weight distribution and gives the wrong advice to the user for every product. The DSAG does not. Next, we relax the requirement that each category have one extremely accurate reviewer to understand how the two advice givers predict as the "best" reviewer becomes less obvious.

Changing Weight Distributions

The greater the difference in weight distributions across different categories, the worse the MAG predicts. We have shown that the MAG can perform significantly worse than the DSAG in this situation. We will now show how the DSAG and MAG predict equally as the reviewers' weights in each category converge to the same distribution. In other words, in the previous example, it is 100% likely that reviewer i will predict correctly in for products in category i and there is a 0% chance that any other user will be correct for that product. Now, we create a review matrix based on a probability p that reviewer i is the "best" reviewer for category i in the following way:

$$R_{ij} = \begin{cases} 5 & (p \wedge \text{ product j in category i}) \\ 1 & ((1-p) \wedge \text{ product j in category i}) \\ 5 & ((1-p) \wedge \text{ product j not in category i}) \\ 1 & (p \wedge \text{ product j not in category i}) \end{cases}$$

With probability p, the reviewer i gives review 5 and the rest of the reviewers give review 1. Otherwise, some other reviewer gives review 5 and the rest give review 1. As it becomes more likely that the reviewer that gives 5 is random, the weight distribution for each category becomes more uniform. The MAG's weight distribution is uniform in all cases as before. It is important to note that if the "best" reviewer is chosen any other way than uniform random, both algorithms would perform better than random because there is a reviewer with higher weight.

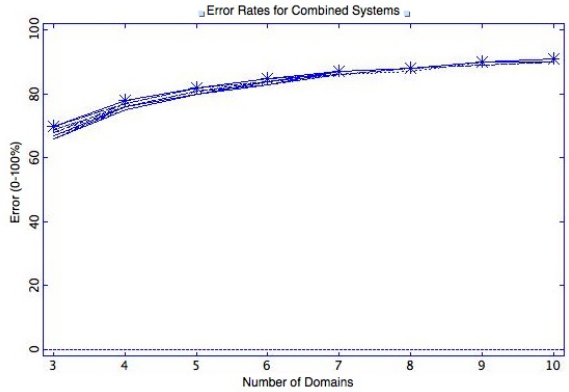


Figure 1: The MAG's % Error over 1000 products consistently is high because of the uniform random weight distribution. X-axis: number of categories and reviewers. Each line: different probability p

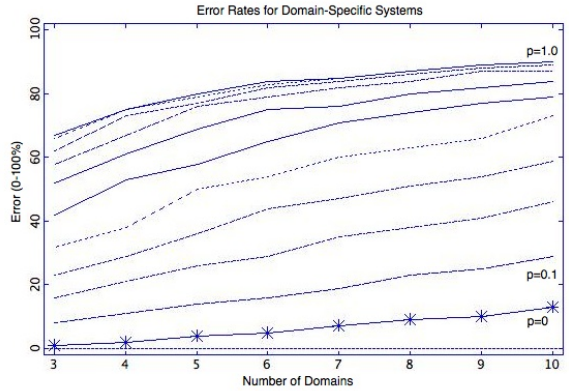


Figure 2: The DSAG's % Error over 1000 products gets worse as the weight distribution becomes uniform random. X-axis: number of categories and reviewers. Each line: different probability p (p=0 on bottom, p=1 on top)

Figures 1 and 2 show the error rates of the MAG and DSAG respectively as we vary $0 \leq p \leq 1$ in increments of .1 and the number of categories from 3 to 10. As the number of categories increases, there are more weight distributions that must each converge, increasing the error rate overall until they do. When $p = 0$, the review matrix is the same as the DSAG example above and the DSAG again has perfect accuracy after converging while the MAG is almost always wrong. As p approaches 1, the domain-specific weights become more uniformly random. However, as we varied p, the mediated weight distribution over the reviewers was consistently uniformly random and the error rate was consistently high. Because it is not holding category information, the MAG doesn't find a difference between truly random ($p = 1$) and domain-correlated ($p = 0$) reviews and predicts the same way each time. The DSAG also has a high error rate as the weight distributions become more uniform. When the MAG and DSAG weight distributions become more similar (to uniform or otherwise), the DSAG and MAG predict the same. The DSAG performs better than the MAG until the distributions are more than 50% similar.

Very Sparse Review Matrices

The previous tests assumed that reviewers gave reviews for all products and all categories. However, this is a strong assumption. Now we relax this assumption and look at much sparser review matrices. For simplicity, we assume that only a single reviewer provides reviews in each category, although this can easily be scaled up to include disjoint *sets* of reviewers. Here, we will also assume that the single reviewer has similar reviews as the user and both give reviews of 5, although we could easily assume that a few of many reviewers are similar. We define the entries of the review matrix in the following way:

$$R_{ij} = \begin{cases} 5 & \text{product j in category i} \\ - & \text{product j not in category i} \end{cases}$$

The DSAG in this case only has a single reviewer to assign a weight for each category and gives it full weight. The MAG, however, must combine these reviewers together into a single weight distribution. Because the products are uniformly distributed in the categories, the reviewers are also uniformly weighted. However, unlike the previous examples where the MAG had a high error rate, the MAG performs equally as well as the DSAG here. Because the MAG only has one review in the product column to compare against, it always picks that review to follow and is 100% accurate - the same as the DSAG.

We have shown in synthetic, simplified review matrices that the MAG can converge on the weight distribution faster than the DSAG with less data and that the DSAG is more accurate when the categorical weight distributions are different. However, we have also shown that the two advice givers perform equally when the weight distributions are the same across categories or when the review matrix is sparse and there are disjoint sets of reviewers for each category. Next, we analyze the properties of actual recommender systems to understand how the two advice givers perform in these real conditions with real data sparsity and real reviewers. It is unclear whether the advice givers even perform the same for two different users of the same recommender system. We use these results to determine how the tradeoff between data and weight distributions affects the accuracy of predictions for different users.

Advice Giver Evaluation - Real Data

To understand how often data sparsity affects the accuracy of the DSAG for users, we built two recommender systems using real data sets from popular recommender websites. We compare the accuracy of the MAG to the DSAG across all users and for each user, and evaluate how the weights of the reviewers affect the success of the MAG. First, we describe the recommender systems' review matrices R.

Experimental Method

The reviewers and advice giver provide possible values $V = \{1, 2, 3, 4, 5\}$ to the user. For each recommender system, the MAG and DSAG provide predictions for the same randomly chosen users. We use the users' reviews of products as the truth for accuracy calculations and remove those reviews

from R. In order to provide both advice givers with the highest chance of convergence, both were given the users' reviews for every product immediately after the prediction. We now present the individual characteristics of three recommender system data sets and the users in those systems.

The 2007 Netflix data set contains over 100 million movie ratings for over 480 thousand Netflix customers from over 17000 movie titles (Netflix 2007). The DSAG used movie genres as domains. Because Netflix did not include genre information in the data set, we cross-referenced movie titles with movie genres and obtained the set the DSAG used: Science Fiction, Fantasy, Special Interest, Kids/Family, Comedy, Western, Action/Adventure, Musical/Performing Arts, Drama, Documentary, and Foreign. Only the movies with a single genre were used, which resulted in a smaller data set of $N = 400$ movies, over 100,000 reviewers, and over 4 million reviews. Approximately 1% of the review matrix R was filled.

The Yahoo! Research Webscope Music review data set contains over 717 million reviews by $M = 1.8$ million Yahoo! Music reviewers for $N = 136{,}000$ songs collected from 2002 to 2006 (Webscope 2008b). Each reviewer was guaranteed to have reviewed at least 30 songs and each song was reviewed by at least 20 reviewers, sparse in comparison to the total number of songs and reviewers. The DSAG uses 20 main genres of music provided in the data set. For each new user, there were an average of 100,000 reviewers giving reviews for their songs.

The final General Products data set was collected from a popular online shopping website (Leskovec, Adamic, and Huberman 2006). Over two years of data collection, 3,943,084 reviewers made 15,646,121 reviews for 548,523 products. 5813 products were discontinued and were thus removed from the set. The DSAG used the ten product categories defined in the dataset (*i.e.,* Book, DVD, Electronics). For each recommendation, the dataset provides the ID of the reviewer, the numerical recommendation as defined above, the date that the recommendation was given, and the product that was reviewed. On average, users made recommendations for about 100 products, with one user recommending over 12000 and each product had between 3 and 2500 recommendations. Approximately .007% of the review matrix is filled.

Results

To evaluate the accuracy of each advice giver, we compare the mean squared error (MSE) of the advice givers' predictions to the users' reviews to capture the distance between the values. We test whether there is a difference between the MSEs of the MAG and DSAG using an ANOVA (Analysis of Variance). 50 users from Netflix and the General Products and 20 users from Yahoo! Music were randomly chosen from the sets to test. We divide this analysis up by the sparsity of the dataset - General Products are sparse, while Netflix and Yahoo! are not as sparse.

Results for Very Sparse Dataset For the General Product data set, our experimental results show there was no significant difference (MAG ΔMSE = .47, DSAG ΔMSE = .53) ($F = .59, df = 1, p > .05$) between the DSAG and MAG. In other words, there is no increase in accuracy as we combine the domain-specific advice givers for each product genre together into a single mediated advice giver. While it is expected that users of multiple systems make reviews for all of them so combining the systems results in more data, we found that users tended to focus their reviews on a specific category of products instead of reviewing products in all categories.

Although the entire user set was shared for all of the domains in the General Product dataset, the set of users that gave reviews for products in one category most likely did not give reviews in any other category. If the most similar users (with the highest weight) in each domain are disjoint, then both DSAG and MAG give the same predictions because the relative weights of the reviewers in each category are the same for both advice givers. We found that the overall performance does not change because given the sparse data, each reviewer only has one accurate weight in one category - the one they provide reviews in. When the weight distributions of the DSAG are combined into the single MAG distribution, the single category weight is carried into the new distribution (and normalized) without the need to average multiple weights together because the weight in the rest of the categories have not changed from the initialized value. The MAG weight distribution is the same as, and not better than, each DSAG.

Results for Other Datasets For the Yahoo! and Netflix data sets, our experimental results show that there is no statistically significant difference between the MSEs of the MAG (MSE=2, std. dev. = .5) and DSAG (MSE=1.8, std. dev = .3) for all users combined ($F = 2.75, df = 1, p > 0.05$). We do find significant differences in the models for individual users. For half of the users in both data sets, the DSAG had a statistically significant lower MSE (ΔMSE = .5, std. dev = .4) than the MAG ($F = 7.14, df = 1, p < 0.01$), but for the other half of the users, the MAG was more accurate (ΔMSE = .25, std. dev. = .2) although this was not statistically significant.

We found (as expected) that the MAG has a higher accuracy than the DSAG when there were not enough products reviewed by the user for the DSAG's domain-specific weights to converge. The DSAG provides more accurate predictions when different reviewers have the highest weight in different domains (See Example in Section 2). Additionally, users for which the DSAG made better predictions typically focused their product requests and reviews in a few domains and did not require all domains, allowing those domain weights to converge quickly. Because half of the users benefitted from each advice giver, neither advice giver should be used to make predictions for all users. We propose an online selection algorithm to autonomously determine which advice giver should make predictions for each user.

We have seen in both synthetic and real recommender system data that when review matrices are very sparse, both advice givers perform the same because the data is both sparse *and* the weight distributions are disjoint. Next, we will an-

alyze the number of categories a reviewer must provide reviews in, in order to see a difference in accuracy between the DSAG and MAG advice givers. Then, assuming reviewers provide enough reviews so that the data is not as sparse, we propose two online selection algorithms to autonomously determine which advice giver should make predictions for each user.

Very Sparse Data

We have shown that in very sparse matrices, there may not be any overlap in reviewers across categories. In other words, reviewers only review products in a single category. The MAG and DSAG produce different weight distributions but the same predictions with the same accuracy. Now, we will show how the MAG is affected as reviewers review products in more categories and as we can collect more data to overcome the extreme sparseness of the review data. We will use the same assumptions as before. The user always gives review 5, so the reviewer that reviews a product with value 5 should have the highest weight.

Different Weight Distributions

First, instead of completely disjoint sets of reviewers for each category, our reviewers review products in a set of categories. Only a single reviewer is correct in each category but several other reviewers also give incorrect reviews. More concretely, we define the review matrix in the following way:

$$R_{ij} = \begin{cases} 5 & \text{product j in category i} \\ 1 & \text{product j in categories (i+1)\%m - (i+k)\%m} \\ - & \text{otherwise} \end{cases}$$

Reviewer i is correct for category i and gives reviews far from the user's preference for categories (i+1)%m to (i+k)%m. The value k is the number of reviewers that provide reviews for each product. By varying k, we can understand how the MAG is affected as the amount of overlap increases in the categories that reviewers review for. We notice that $k = 0$ means that only a single reviewer reviews products in a single category. Also, $k = m$ means that all reviewers provide reviews for all products. We are interested in how the MAG behaves for values between 0 and m. We know that although it may take the DSAG longer to converge on the categorical weight distributions, it will always have 100% accuracy.

Figure 3 shows the error rate (y-axis) for the MAG and DSAG with $m = \{1, ..., 10\}$ (x-axis) and with $k = \{1, ..., 10\}$. Each line represents a different k. Note that there cannot be more than m reviewers overlapping at a time, so the line for k starts at m. The line for 3 overlapping reviewers starts at 100% error rate at $m = 3$ and drops a little before converging at a high error rate. Because all reviewers are "best" for one category, the MAG converges to a uniform distribution. A majority of the reviewers give reviews that conflict with the user, but the MAG follows the majority so it predicts incorrectly nearly all the time depending on

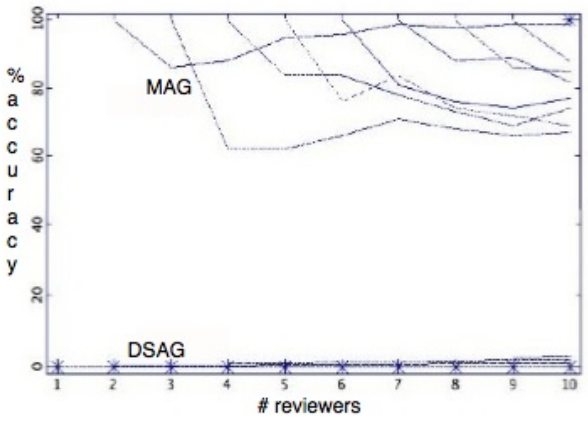


Figure 3: The DSAG has almost 0% error no matter how many reviewers review each product. The MAG has perfect accuracy when the reviewers in each category are disjoint $(k = 1)$. However, it is almost always wrong for $k > 1$ and any amount of overlap with reviewers.

the order of products. The DSAG always finds the correct weight distribution and maintains a 0% error over all k.

There is no statistical significance between the error rates of the ks. As we increase k, the error is nearly constant. However this error is expected because the weight distributions are so different for each category. Next, we evaluate the MAG performance on categorical weight distributions that are similar but have sparse reviewer data.

Similar Weight Distributions

We know that the MAG converges faster than the DSAG when the categorical weight distributions are similar and all reviewers always give reviews. Now we create a new review matrix such that there are a few (*e.g.,* 2) "best" reviewers for a user and the rest give opposing reviews:

$$R_{ij} = \begin{cases} 5 & (p \wedge i = 1) \\ 5 & ((1-p) \wedge i = 2) \\ 1 & (q \wedge i \neq 1) \\ - & \text{otherwise} \end{cases}$$

With some probability p, reviewer 1 gives the best response and otherwise reviewer 2 gives the best response. Then the rest of the reviewers each give a response with some other probability q and otherwise give no response. These distributions do not depend on the category, so the DSAG would have to find this same distribution for each category. The MAG finds it just once.

If either reviewer 1 or reviewer 2 always provides a review for a product, the MAG can find these best reviewers after getting reviews from each of them only a few times. Because these reviewers are always correct with respect to our user, a MAG that always trusts them will also always be correct. Although these reviewers never appear at the same time, the MAG still has one reviewer that has high weight to

use. The rest of the reviewers have low weight and are not included in the prediction. If there are products for which neither of the "best" reviewers provide reviews, there is no way for the MAG to produce a good answer. The MAG can be as accurate and converge on the reviewer weights faster than the DSAG when the similarity distributions are the same and reviewers do provide reviews for products in multiple categories.

We have shown that the MAG and DSAG are each more accurate for some users, depending on the products they choose from the review matrix. In very sparse review matrices where reviewers provide reviews in only a single category, both the MAG and DSAG have the same accuracy. If we can increase the amount of data to get reviewers to review products in multiple categories, both the DSAG and MAG can perform well under some conditions. Because it is not known which advice giver will be more accurate when a user joins a recommender system, the recommender system must calculate predictions of both advice givers as the user requests reviews and provides reviews to determine which is more accurate. We present two online selection algorithms for the recommender system to use in determining which advice giver (DSAG or MAG) makes the best predictions for each user.

User-Dependent Online Selection Algorithm

First, we show a user-dependent online selection algorithm (UdOS) which picks the single best advice giver for each user. The UdOS Algorithm (Algorithm 3) first initializes a new DSAG and MAG for the new user and sets their accuracies to 0 (Steps 1-2). Because the MAG provides better predictions when there are fewer data points available, the MAG's predictions are provided to each user to start (Step 3). Additionally, the algorithm requires that the domain-specific weights converge before making a decision. It has been shown that the weighted majority algorithm requires approximately $\lceil log(M) \rceil$ user reviews to converge (Littlestone and Warmuth 1994). However, the advice givers actually require at least one review from each advice giver find the most accurate weights. Thus, the algorithm takes linear time $O(M)$ in the number of reviewers to converge. The count $num\_usr\_reviews$ of the current number of reviews is initialized to 0 (Step 4).

Until the recommender system has received enough user reviews, the UdOS algorithm makes predictions for the user with the current most accurate advice giver (Steps 5-17). The user requests a product p_j to be predicted by the recommender system (Step 6). The algorithm gets both the MAG and DSAG predictions (Steps 7-8), but gives the user only the prediction from the current best advice giver (Step 9). If the recommender system receives the user's actual review a_j, the algorithm increases the count of the user reviews, and recalculates the best advice giver using the new accuracy (Steps 10-16). After enough reviews have been collected from the user, UdOS maintains only the best advice giver, which continues to reweigh reviewers (Steps 18-21).

Using the UdOS algorithm for each user, the recommender system can decide whether the data sparsity or reviewers' weights affect the prediction accuracy more and choose the best advice giver. Although it is less computationally efficient to calculate the predictions from both the DSAG and MAG, because the algorithm picks the single best advice giver, the recommender system provides better predictions for all users. After making the selection, it saves memory and computation by maintaining only the best one.

Algorithm 3 User-Dependent Selection Algorithm

1: $dsag \leftarrow$ new $DSAG(u)$, $mag \leftarrow$ new $MAG(u)$
2: $acc_{MAG} \leftarrow 0$, $acc_{DSAG} \leftarrow 0$
3: $BestAG \leftarrow mag$
4: $num\_usr\_reviews \leftarrow 0$
5: **while** $num\_usr\_reviews < M$ **do**
6: $p_j \leftarrow getProductRequest()$
7: $pred_{MAG} \leftarrow mag.predict(p_j)$
8: $pred_{DSAG} \leftarrow dsag.predict(p_j)$
9: **print** $pred_{BestAG}$
10: **if** receive a_j **then**
11: $num\_usr\_reviews$ ++
12: $acc_{MAG} \leftarrow calcAcc(mag)$
13: $acc_{DSAG} \leftarrow calcAcc(dsag)$
14: $mag.update(a_j)$, $dsag.update(a_j)$
15: $BestAG \leftarrow$ if $(acc_{MAG} < acc_{DSAG})$ $?mag$: $dsag$
16: **end if**
17: **end while**
18: **loop**
19: $p \leftarrow getProductRequest()$
20: **print** $BestAG.predict(p)$
21: **end loop**

User- and Category-Dependent Online Selection Algorithm

Our second selection algorithm picks the best advice giver for each category (UCdOS). If the DSAG converges for one or a few categories because the user focuses on those categories, this algorithm can pick the DSAG for those that have converged and continue to use the MAG for all other categories. The UCdOS Algorithm (Algorithm 4) first initializes a new DSAG and MAG for the new user and sets their accuracies to 0 (Steps 1-5) as in the UdOS algorithm.

Until recommender system has received enough user reviews, the UCdOS algorithm makes predictions for the user with the current most accurate advice giver for each category (Steps 5-18). The user requests a product p_j in category c_k to be predicted by the recommender system (Steps 6-7). The algorithm gets both the MAG and DSAG predictions (Steps 8-9), but gives the user only the prediction from the current best advice giver for category c_k (Step 10). If the recommender system receives the user's actual review a_j, the algorithm increases the count of the user reviews, and recalculates the best advice giver using the new accuracy (Steps 11-17). After enough reviews have been collected from the user, UCdOS maintains only the best advice giver for each category, which continues to reweigh reviewers (Steps 19-23).

Using the UCdOS algorithm for each user, the recom-

Algorithm 4 User- and Category-Dependent Selection Alg.

```
1: dsag ← new DSAG(u), mag ← new MAG(u)
2: acc_MAG ← 0, acc_DSAG ← 0
3: BestAG ← mag
4: num_usr_reviews ← 0
5: while num_usr_reviews < ⌈log(M)⌉ do
6:    p_j ← getProductRequest()
7:    c_k ← p_j.getCategory()
8:    pred_MAG ← mag.predict(p_j)
9:    pred_DSAG ← dsag.predict(p_j)
10:   print pred_BestAG[c_k]
11:   if receive a_j then
12:      num_usr_reviews ++
13:      acc_M ← calcAcc(mag)
14:      acc_DS ← calcAcc(dsag)
15:      mag.update(a_j), dsag.update(a_j)
16:      BestAG[c_k] ← if (acc_M < acc_DS) ?mag : dsag
17:   end if
18: end while
19: loop
20:   p ← getProductRequest()
21:   c ← p.getCategory()
22:   print BestAG[c].predict(p)
23: end loop
```

mender system can decide for each category whether the data sparsity or reviewers' weights affect the prediction accuracy more and choose the best advice giver on a category specific basis. This allows for more flexibility in the products the user picks. The UdOS algorithm requires that the user pick products from all categories uniformly in order to determine whether the DSAG or MAG is better overall for all categories. In the UCdOS algorithm, even if users do not pick products uniformly across categories, the most appropriate advice giver is chosen. If a user does not request products in some categories, there is less data to train from and the MAG would produce better results. When a weight distribution for a category has converged, the DSAG is more appropriate to use and the UCdOS algorithm uses that.

The UCdOS is less computationally efficient than the UdOS, because it must maintain the DSAG and MAG until all of the DSAG weight distributions converge, but if it means that it provides better predictions than any other algorithm it may be worth it.

Validation

The UdOS and UCdOS algorithms will choose the advice giver with the highest accuracy after enough products are reviewed by the user. Ideally, the accuracy of the predictions from the algorithm will be as good as the best advice giver even through the selection process in order to maintain user satisfaction with the recommender system. In order to validate that the UdOS and UCdOS algorithms both pick the better advice giver and maintain high accuracy, we tested them against the MAG and DSAG advice givers on a fourth recommender system data set. We calculate accuracy over time to understand how both selection algorithm perform and show that they finds the best advice giver for each user while minimizing errors.

Experimental Method

The Yahoo! Movies data set contains 7,642 users, 11,915 movies and 211,231 reviews (Webscope 2008a). We use the same movie genres for domains as the Netflix movies data set. Only the movies with a single genre were used which resulted in a smaller data set of 9,000 reviewers for 1000 movies. Only 0.23% of the review matrix R was filled. Each product was reviewed by at least two reviewers.

We use the same recommender system setup as the experimental results above. The advice givers and selection algorithms provide possible values $V = \{1, 2, 3, 4, 5\}$ for the same randomly chosen users. The advice givers received the users' reviews after every product prediction and use the reviews, which were removed from R, to reweigh reviewers. We report the MSE instead of accuracy because it better represents the error distance between the prediction and users' reviews. Additionally, because we did not have enough data to wait for all of the M reviews, we used $logM$ as the parameter to wait before converging on the best advice giver so that we could report a best advice giver for each user. In general, it became obvious which advice giver to choose after waiting for only $logM$ reviews for this data set although for some users it was not sufficient.

We found that the $calcAcc$ function should place more weight on the later predictions and less weight on the earlier predictions because the advice giver was more likely to be incorrect with fewer user reviews when determining the best advice giver. We used a step function as follows although we found that any linear function or step function with a different span produced the same results.

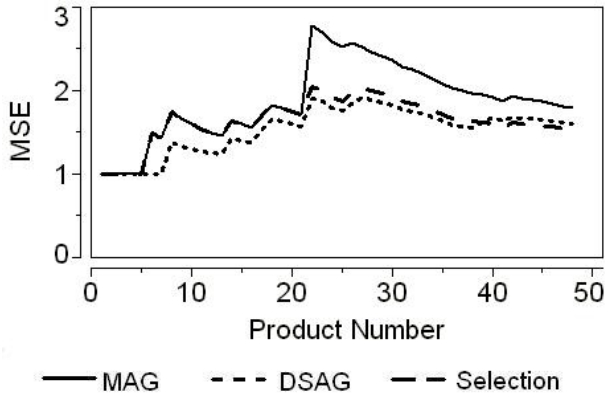
$$calcAcc(ag) = avg(\text{accuracy recent 10 predictions})$$

If the DSAG makes poor predictions for new domains at the beginning of the product set, the UdOS and UCdOS will always determine that it has a lower accuracy. We use both the cumulative (unweighted) and weighted MSEs in the results. Twenty users were chosen at random as test users. Each user requested predictions for between 20 and 130 products. The MAG, DSAG, and UdOS and UCdOS algorithms' predictions were recorded for each user and used to calculate each unweighted MSE (see Tables 3 and 4).

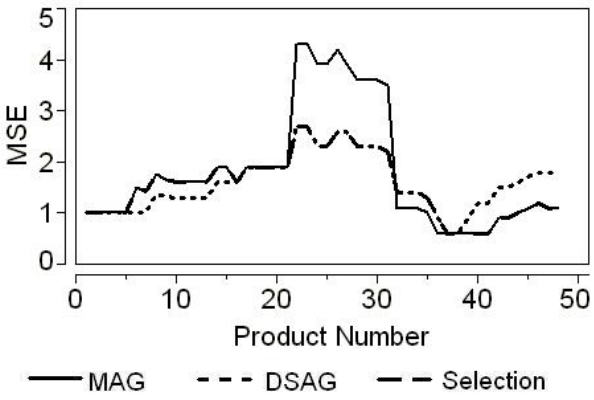
UdOS Results

Overall, thirteen users received better predictions from the DSAG and 6 received better predictions from the MAG. One user received equivalent predictions from both advice givers. Our results show that for all 20 users the UdOS algorithm picked the best advice giver. Furthermore, the UdOS algorithm's error was no worse than the worst advice giver and at times can be better than the best advice giver. We will now describe in more detail how the UdOS algorithm chose the advice givers for some users.

For User 1, data sparsity affected the predictions from the DSAG and he received better predictions from the MAG, so the UdOS algorithm continued using the MAG for the duration of the product set. User 19 received better predictions



(a) The cumulative MSE for User 8 over time.



(b) The weighted MSE for User 8 over time.

Figure 5: The UdOS used the DSAG for User 8 on products 5-32 when the MAG was providing worse predictions. After product 32, the MAG provided better predictions and the UdOS switched back to using it.

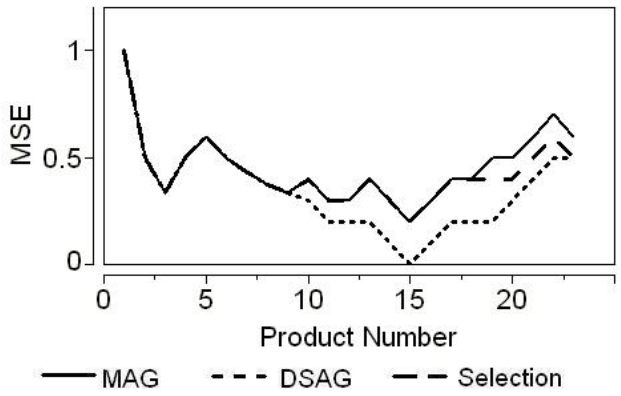


Figure 4: For User 19, the UdOS algorithm uses the MAG advice until it has seen enough data to conclude that the DSAG is better and then switches to use its predictions (product 18).

User	MAG	DSAG	UdOS	Selection
1	**1.09**	1.29	1.09	MAG
2	**1.29**	1.40	1.38	MAG
3	**1.35**	1.49	1.35	MAG
4	**1.32**	1.41	1.32	MAG
5	**1.27**	1.36	1.27	MAG
6	**2.48**	2.48	2.48	MAG**
7	1.09	**1.08**	1.08	MAG*
8	1.78	**1.59**	1.51	MAG*
9	**1.04**	1.14	1.03	DSAG*
10	0.95	**0.92**	0.95	DSAG
11	0.51	**0.38**	0.51	DSAG
12	0.54	**0.49**	0.51	DSAG
13	1.08	**0.58**	0.61	DSAG
14	1.06	**0.81**	1.03	DSAG
15	1.03	**0.97**	1.03	DSAG
16	0.80	**0.73**	0.76	DSAG
17	0.67	**0.63**	0.63	DSAG
18	1.07	**0.89**	1.07	DSAG
19	0.50	**0.38**	0.46	DSAG
20	1.44	**1.37**	1.39	DSAG

Table 3: Six users received better predictions with lower cumulative MSEs (bold) from the MAG while 13 received better predictions from the DSAG.

from the DSAG than the MAG, because the user's similar reviewers were different for each product domain. The UdOS algorithm recognized that the DSAG had a lower MSE (See Figure 4) and switched to using the DSAG's predictions at product 18. After the switch, the UdOS cumulative MSE converged towards that of the DSAG.

The algorithm picked the DSAG for other users like 10, but the switch occurred late enough that the cumulative MSE did not start converging towards the DSAG's MSE. Because the cumulative MSE was the same for both the DSAG and MAG for User 16, it didn't matter which advice giver the UdOS picked (marked with ** in Table 3). As a result, it continually used the MAG that it started with.

The UdOS algorithm picked the advice giver with higher overall MSE for three users (marked with * in Table 3). User 8, for example, received better predictions from the UdOS algorithm than either the MAG or DSAG could provide alone because the UdOS switched back and forth between the two advice givers before deciding which to use. The MSE of the DSAG is lower than the MAG's because the MAG poorly predicted some products in the middle of the set (specifically products 20-30) (See Figure 5(a)). However, once the MAG's weight distributions identified the most similar reviewers (product 32), its predictions were better than the DSAG's and the UdOS algorithm picked the MAG (See 5(b)).

UCdOS Results

Overall, nine users received the same predictions as the UdOS, picking either one advice giver or the other for all of the product categories the user requested (See Table 4, no *). Note that this does not mean that the DSAG or MAG

was chosen for all categories, just that it picked for all categories that there was data for. If a new product from a new category was queried, while the UdOS would have to use the same advice giver as it picked for the other categories, the UCdOS could use the MAG until enough products in that category were reviewed. This feature means that the UCdOS could be more accurate using the MAG while the DSAG's weight distribution for the new category converges. However it could be less accurate if the MAG distribution is very different than the category weight distribution.

Six of the users received worse predictions from the UCdOS compared to the UdOS, but still at least as good as the worse advice giver (See Table 4, *). In these cases, the MAG's single distribution was different from the DSAG's category distributions. The UCdOS used the MAG for each category until the DSAG converged. The MAG produced worse predictions in the UCdOS than if the DSAG was chosen starting from a uniform distribution in the UdOS. This is a particularly good instance of our initial example that the MAG's poor results when the DSAG's categorical distributions are very different. The UdOS chooses the DSAG as soon as there is any indication that the weight distributions are different for different categories. Because it takes time for each of the DSAG's to converge, it is worse for the UCdOS to use the MAG before the convergence even though it has an indication that the DSAG's weights are different than the MAG's. In this case, waiting ensure the selection algorithm picks the right advice giver for each category actually makes the prediction accuracy worse.

Five reviewers received better predictions than the UdOS, including several who received better predictions than either the DSAG or MAG could provide alone (See Table 4, **). These users had the best benefit from the data sparsity versus weight distribution tradeoff. The UCdOS was able to use the DSAG for the categories that the user queried often and the MAG for the other categories. The DSAG's category weight distributions were similar to the MAG's distribution for some categories and different in others. The UCdOS did not have to pick a single advice giver and could take advantage of the MAG instead of waiting for the category weight distribution to converge to the same values. This sped up the selection process, allowing the UCdOS to be more accurate for longer than either of the two advice givers alone.

User	MAG	DSAG	UCdOS
1	1.09	1.29	1.09
2	1.29	1.40	1.27*
3	1.35	1.49	1.46**
4	1.32	1.41	1.32
5	1.27	1.36	1.27
6	2.48	2.48	2.48
7	1.09	1.08	1.09**
8	1.78	1.59	1.78**
9	1.04	1.14	0.99*
10	0.95	0.92	0.89*
11	0.51	0.38	0.51
12	0.54	0.49	0.51
13	1.08	0.58	1.05**
14	1.06	0.81	1.06**
15	1.03	0.97	1.03
16	0.80	0.73	0.76
17	0.67	0.63	0.63
18	1.07	0.89	0.89*
19	0.50	0.38	0.41*
20	1.44	1.37	1.41**

Table 4: Five users (*) received better predictions from the UCdOS while six users received better predictions from the UdOS (**).

Discussion

Both of the user selection algorithms pick the best advice givers with the knowledge they have. The UdOS algorithm picks a single advice giver to use for the rest of the predictions. Once it finds that two different categories have different weight distributions and the DSAG is performing better, it switches to using it. The algorithm makes the assumption that if there are two weight distributions that are different, that all will be different. More importantly, it makes a strong tradeoff that the different distributions are more important than the data sparsity problem. If there are different distributions, the algorithm must find each of the categorical weight distributions and cannot take advantage of the MAG while they converge.

The UCdOS algorithm picks an advice giver for each category. It does not make the assumption that all weight distributions are different, but does assume that the distributions are either like the MAG distribution or not. It can take advantage of the MAG while each of the DSAG distributions are converging, which greatly improved the accuracy of the predictions for five users compared to the UdOS. However, we found that sometimes the MAG is a detriment and using it makes the selection algorithm predict worse than the UdOS when the MAG distribution is different from the category weight distribution. Overall, both selection algorithms do well for all of our test users and there is no clear better algorithm to choose.

These selection algorithms do pick the best advice giver, but still require that the weight distributions either be completely different from each other or the same as the MAG. It is possible that high weight reviewers in DVDs have the same high weight in books because the user likes mysteries in both categories but have very different taste in clothes. One other selection algorithm that could be tested in future work would be one that tests whether any of the converged category weight distributions perform well for a new category. This algorithm would take advantage of all the previous work done to converge the distribution and speed up the convergence of a new distribution making the algorithm more accurate earlier in the product queries. It would be more costly, however, in the space required to hold all of the possible distributions.

Conclusion

In this work, we first presented an overview of how advice givers weigh reviewers and make predictions for users. We presented the Mediated (MAG) and Domain-Specific (DSAG) advice givers that weigh the tradeoffs between data sparsity and reviewers' weights differently and affect predic-

tion accuracy in recommender systems. Although the DSAG provides more accurate predictions when there are sufficient reviews, using a MAG was recently suggested when review data is too sparse. We used three real recommender data sets to show that both advice givers provide accurate predictions to some users. The DSAG provides more accurate predictions to users when different reviewers were weighed highly in different domains while the MAG is more accurate when the review matrix and user reviews are sparse.

We presented our User-Dependent Online Selection Algorithm and User- and Category-Dependent Online Selection algorithm as two online methods for recommender systems to decide which advice giver is more accurate for each user. The algorithms wait for the reviewers' weights to converge before picking the best advice giver. We validated our findings on a fourth recommender data set and demonstrated that the UdOS algorithm successfully picks the better advice giver to provide the most accurate predictions possible for each user. We conclude that a recommender system that uses the UdOS or UCdOS algorithms make better predictions for all users than one that uses only one of the two advice givers.

References

Adomavicius, G., and Tuzhilin, A. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowledge and Data Engineering* 734–749.

Auer, P.; Cesa-Bianchi, N.; Freund, Y.; and Schapire, R. 1995. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *FOCS*, 322–331.

Berkovsky, S.; Kuflik, T.; and Ricci, F. 2007a. Cross-domain mediation in collaborative filtering. In *User Modeling*, 355–359.

Berkovsky, S.; Kuflik, T.; and Ricci, F. 2007b. Distributed collaborative filtering with domain specialization. In *Recommender Systems*, 33–40.

Blum, A., and Mansour, Y. 2005. From external to internal regret. In *COLT*, 621– 636.

Burke, R. 2002. Hybrid recommender systems: Survey and experiements. *User Modeling and User-Adapted Interaction* 12(4):331–370.

Freund, Y.; Schapire, R.; Singer, Y.; and Warmuth, M. 1997. Using and combining predictors that specialize. In *STOC*, 334–343.

Leskovec, J.; Adamic, L.; and Huberman, B. 2006. The dynamics of viral marketing. In *ACM Conf. on Electronic Commerce*, 228–237.

Littlestone, N., and Warmuth, M. 1994. The weighted majority algorithm. *Information and Computation* 212–261.

Nakamura, A., and Abe, N. 1998. Collaborative filtering using weighted majority prediction algorithms. In *International Conference on Machine Learning*, 395–403.

Netflix. 2007. The netflix dataset and prize.

Umyarov, A., and Tuzhilin, A. 2007. Leveraging aggregate ratings for better recommendations. In *Recommender Systems*, 161–164.

Webscope, Y. R. 2008a. Movie user ratings of movies and descriptive content information v1.0.

Webscope, Y. R. 2008b. Music user ratings of songs with song attributes v1.0.

Using Gaussian Spatial Processes to Model and Predict Interests in Museum Exhibits

Fabian Bohnert, Ingrid Zukerman, and Daniel F. Schmidt
Faculty of Information Technology, Monash University
Clayton, VIC 3800, Australia
{fabianb,ingrid,dschmidt}@infotech.monash.edu.au

Abstract

This paper adapts models from the area of spatial statistics to the task of predicting a user's interests (i. e., implicit item ratings) within a recommender system in the museum domain. We develop a model based on Gaussian spatial processes, and discuss two ways of computing item-to-item distances in the museum setting. Our model was evaluated with a real-world dataset collected by tracking visitors in a museum. Overall, our model attains a higher predictive accuracy than nearest-neighbour collaborative filters. In addition, the model variant using physical distances outperforms that using distances computed from item-to-item similarities.

1 Introduction

Spatial processes (*random fields*) are a subclass of *stochastic processes* which are applied to domains that have a geospatial interpretation, e. g., [Diggle *et al.*, 1998; Banerjee *et al.*, 2004]. They are typically used in the field of spatial statistics to model spatial associations between a set of observations made at certain locations, and to predict values at locations where no observations have been made. This paper applies such models to the prediction of a user's interests or item ratings in recommender systems (RS). We develop our *Spatial Process Model (SPM)* by adapting a Gaussian spatial process model to the RS scenario, and demonstrate our model's applicability to the task of predicting implicit ratings in the museum domain. The use of spatial processes requires a measure of distance between items in addition to users' ratings. This measure, which is non-specific (e. g., it may be a physical or a conceptual distance), can be readily obtained in most cases. For example, distances could be computed from feature vectors representing the items (similarly to content-based RS), from item-to-item similarities (similarly to item-to-item collaborative filtering [Sarwar *et al.*, 2001]), or from physical distance. In this paper, we explore the latter two measures.

Our application scenario is motivated by the need to automatically recommend exhibits to museum visitors, based on non-intrusive observations of their actions in the physical space. Employing RS in this scenario is challenging due to (1) the physical nature of the domain, (2) having exhibit viewing times rather than explicit ratings, and (3) predictions differing from recommendations (we do not want to recommend exhibits that visitors are going to see anyway). We turn the first challenge into an advantage by exploiting the fact that physical distances between exhibits are meaningful, enabling the use of walking distance between exhibits to calculate (content) distance. This supports the direct, interpretable application of spatial processes by using a simple parametric Gaussian spatial process model (with the ensuing low variance in parameter estimates), compared to more complex non-parametric approaches, e. g., [Schwaighofer *et al.*, 2005]. The second challenge, which stems from the variable semantics of viewing times (time t for different exhibits could mean interest or boredom), is naturally addressed by *SPM*'s structure. The third challenge can be addressed by (a) using *SPM* to build a model of a visitor's interests in unseen exhibits, (b) inferring a predictive model of a visitor's pathway through the remainder of the museum [Bohnert *et al.*, 2008], and (c) combining these models to recommend exhibits of interest that may be overlooked if the predicted pathway is followed.

SPM was evaluated with a real-world dataset of time spans spent by museum visitors at exhibits (viewed as implicit ratings). We compared our model's performance to that of (1) a baseline model which delivers a non-personalised prediction, and (2) a nearest-neighbour collaborative filter incorporating performance-enhancing modifications, e. g., [James and Stein, 1961; Herlocker *et al.*, 1999]. Our results show that *SPM* significantly outperforms both models.

The paper is organised as follows. In Section 2, we discuss related research. Section 3 describes our domain and dataset. Our spatial processes approach for modelling and predicting exhibit interests is developed in Section 4. In Section 5, we present the results of our evaluation, followed by a discussion in Section 6 and our conclusions in Section 7.

2 Related Research

Recommender systems (RS) are designed to direct users to personally interesting items in situations where the amount of available information exceeds the users' processing capability [Resnick and Varian, 1997; Burke, 2002]. Typically, such systems (1) use information about a user (i. e., a user model) to predict ratings of items that the user has not yet considered, and (2) recommend suitable items based on these predictions. *Collaborative* modelling techniques constitute

one of the main model classes applied in RS [Albrecht and Zukerman, 2007]. They base their predictions upon the assumption that users who have agreed in their behaviour in the past will agree in the future.

The greatest strength of collaborative approaches is that they are independent of any representation of the items being recommended, and work well for complex objects, for which features are not readily apparent. The two main collaborative approaches are *memory-based* and *model-based*. Previous research has mainly focused on memory-based approaches, such as nearest-neighbour models (classic collaborative filtering), e. g., [Herlocker *et al.*, 1999]. The main drawback of memory-based algorithms is that they operate over the entire user database to make predictions. In contrast, model-based approaches use techniques such as Bayesian networks, latent-factor models and artificial neural networks, e. g., [Breese *et al.*, 1998; Bell *et al.*, 2007], to first learn a statistical model in an offline fashion, and then use it to make predictions and generate recommendations. This decomposition can significantly speed up the recommendation generation process.

Personalised guide systems in physical domains have often employed adaptable user models, which require visitors to explicitly state their interests in some form. For example, the *GUIDE* project [Cheverst *et al.*, 2002] developed a handheld tourist guide for visitors to the city of Lancaster, UK. It employed a user model obtained from explicit user input to generate a dynamic and user-adapted city tour, where the order of the visited items could be varied. In the museum domain, the *CHIP* project [Aroyo *et al.*, 2007] investigates how Semantic Web techniques can be used to provide personalised access to digital museum collections both online and in the physical museum, based on models that require an explicit initialisation.

Less attention has been paid to predicting preferences from non-intrusive observations, and to utilising adaptive user models that do not require explicit user input. In the museum domain, adaptive user models are usually updated from a user's interactions with the system, the focus being on adapting content presentation as opposed to predicting and recommending exhibits to be viewed. For example, *HyperAudio* [Petrelli and Not, 2005] dynamically adapted the presented content and hyperlinks to stereotypical assumptions about a user, and to what a user has already accessed and seems interested in. The augmented audio reality system for museums *ec(h)o* [Hatala and Wakkary, 2005] treated user interests in a dynamic manner, and adapted its user model on the basis of a user's interactions with the system. The collected user modelling data were used to deliver personalised information associated with exhibits via audio display. The *PEACH* project [Stock *et al.*, 2007] developed a multimedia handheld guide which adapts its user model on the basis of both explicit visitor feedback and implicit observations of a visitor's interactions with the device. This user model was then used to generate personalised multimedia presentations.

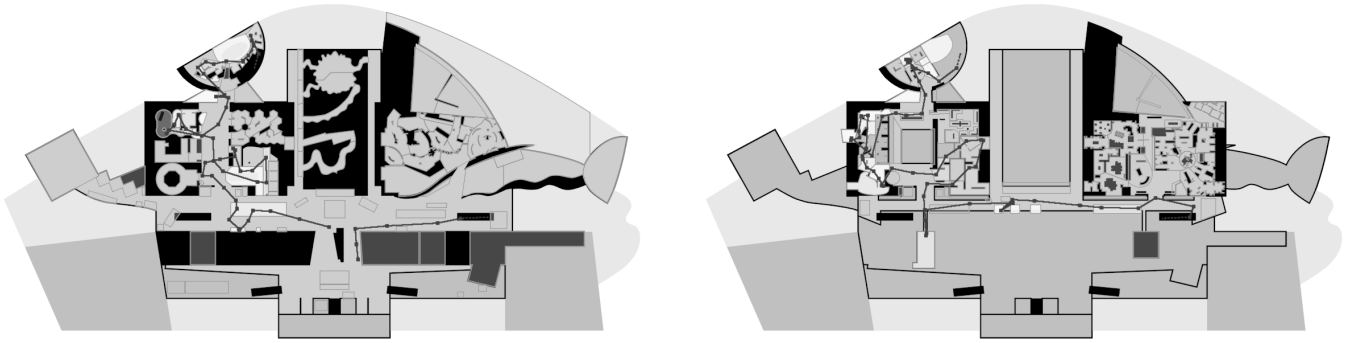
These systems, like most systems in the museum domain, rely on knowledge-based user models in some way, and hence, require an explicit, a-priori engineered representation of the domain knowledge. In contrast, our research investigates non-intrusive statistical user modelling and recommendation techniques that do not require such an explicit domain knowledge representation [Albrecht and Zukerman, 2007].

3 Domain and Dataset

The GECKO project endeavours to develop user modelling and personalisation techniques for information-rich physical spaces, relying on non-intrusive observations of users' behaviour [Bohnert *et al.*, 2008]. Developing such non-intrusive user modelling and personalisation techniques for museums requires datasets about visitor behaviour in the physical museum space (i. e., visitor pathways). Datasets that are suitable for the development phase can be obtained by manually tracking museum visitors. Such a data collection methodology is clearly inappropriate for model deployment, but it facilitates model development by eschewing issues related to technology selection and instrumentation accuracy.

Museums such as Melbourne Museum (Melbourne, Australia) display thousands of exhibits distributed over many separate galleries and exhibitions. Normally, visitors do not require recommendations to travel between individual, logically related exhibits in close physical proximity. Rather, they may prefer recommendations regarding physically separated areas. In order to gather data for assessing predictive models that support appropriate recommendations, we grouped Melbourne Museum's individual exhibits into semantically coherent and spatially confined *exhibit areas*. This task, which was performed with the assistance of museum staff, yielded 126 exhibit areas. Figure 1 depicts the site map of Melbourne Museum showing these exhibit areas, together with one of the visitor pathways we collected.

To obtain our dataset, we manually tracked visitors to Melbourne Museum from April to June 2008, using a custom-made tracking tool running on laptop computers [Bohnert and Zukerman, 2009]. In total, we recorded over 170 visitor pathways. We only tracked first-time adult visitors travelling on their own, to ensure that neither prior knowledge about the museum nor other visitors' interests influenced a visitor's decisions about which exhibits to view. Prior to the data collection, we briefed our trackers on the usage of our tracking software, the layout of the museum, and its digital representation on the site map. Additionally, we clarified what should be considered a viewing event. After the data collection, the visitor pathways were post-processed using a post-processing tool we developed. For instance, we removed tracking events that could not have possibly occurred, e. g., visitor transitions from one end of the museum to the other and back within a few seconds, or transitions outside the museum walls and back. We also removed incomplete visitor pathways, e. g., due to a laptop computer running out of battery, or a visitor leaving unexpectedly. The resulting dataset comprises 158 complete visitor pathways in the form of time-annotated sequences of visited exhibit areas, with a total visit length of 291:22:37 hours, and a total viewing time of 240:00:28 hours. The dataset also contains demographic information about the visitors, which was obtained by means of post-visit interviews conducted by our trackers. In total, we obtained 8327 viewing durations at the 126 exhibit areas, yielding an average of 52.7 exhibit areas per visitor (41.8% of the exhibit



(a) Melbourne Museum – Ground level

(b) Melbourne Museum – Upper level

Figure 1: Visitor pathway visualised on a site map of Melbourne Museum

Table 1: Dataset statistics

	Mean	Stddev	Min	Max
Visit length (hrs)	1:50:39	0:47:54	0:28:23	4:42:12
Viewing time (hrs)	1:31:09	0:42:05	0:14:09	4:08:27
Exhibit areas / visitor	52.70	20.69	16	103
Visitors / exhibit area	66.09	25.36	6	117

areas). Hence, on average 58.2% of the exhibit areas were not viewed by a visitor. This indicates that there is potential for pointing a visitor to relevant but unvisited exhibit areas. Table 1 summarises further statistics of the dataset.

Clearly, the deployment of an automated RS in a museum requires suitable positioning technologies to non-intrusively track visitors, and models to infer which exhibits are being viewed. Although our dataset was obtained manually, it provides information of the type that may be inferred from sensing data (the work described in [Schmidt *et al.*, 2009] links sensory and manually obtained information). Additionally, the results obtained from experiments with this dataset are essential for model development, as they provide an upper bound for the predictive performance of our model.

4 Using Gaussian Spatial Processes to Model and Predict Visitors' Exhibit Interests

In this section, we first describe how we use viewing time to quantify interest in exhibits (Section 4.1), and discuss the applicability of spatial process models [Banerjee *et al.*, 2004] to the prediction of a visitor's interest in exhibits in our RS scenario (Section 4.2). We then propose a model-based collaborative approach based on the theory of Gaussian spatial processes for predicting a visitor's (log) viewing times (viewed as exhibit interests) from non-intrusive observations of his/her (log) viewing times at visited exhibits (Section 4.3).

4.1 From Viewing Time to Exhibit Interest

In an information-seeking context, people usually spend more time on relevant information than on irrelevant information, as viewing time correlates positively with preference and interest [Parsons *et al.*, 2004]. Hence, viewing time can be used as an indirect measure of interest. We propose to use log viewing time (instead of raw viewing time), due to the following reasons. When examining our dataset (Section 3), we found the distributions of viewing times at exhibits to be positively skewed (we use the terms 'exhibit' and 'exhibit area' synonymously in the remainder of this paper). Thus, the usual assumption of a Gaussian model did not seem appropriate. To select a more appropriate family of probability distributions, we used the *Bayesian Information Criterion (BIC)* [Schwarz, 1978]. We tested exponential, gamma, normal, log-normal and Weibull distributions. The log-normal family fitted best, with respect to both number of best fits and average BIC score (averaged over all exhibits). Hence, we transformed all viewing times to their log-equivalent to obtain approximately normally distributed data. This transformation fits well with the idea that for high viewing times, an increase in viewing time indicates a smaller increase in the modelled interest than a similar increase in the context of low viewing times.

4.2 Spatial Statistics in the Context of Our Application Scenario

Spatial statistics is concerned with the analysis and prediction of geographic data [Banerjee *et al.*, 2004]. Utilising spatial processes, the field deals with tasks such as modelling the associations between observations made at certain locations, and predicting values at locations where no observations have been made. The assumption made for spatial processes, that correlation between observations increases with decreasing site distance, fits well with our RS scenario, where viewing times are usually more correlated the more related exhibits are. Hence, by introducing a notion of spatial distance between exhibits to functionally specify this correlation structure, we can use spatial process models for predicting viewing times (i. e., exhibit interests). We use $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n$ to denote the locations of exhibits $i, j \in I = \{1, \ldots, n\}$ in a space providing such a distance measure, i. e., $\|\boldsymbol{s}_i - \boldsymbol{s}_j\|$. For example, $\|\boldsymbol{s}_i - \boldsymbol{s}_j\|$ can be computed from feature vectors representing the items (similarly to content-based RS), from item-to-item similarities (similarly to item-to-item collaborative filtering [Sarwar *et al.*, 2001]), or from physical distance.

In this paper, we explore the two latter options: *Item-to-Item Distance* and *Physical Distance*.

- **Item-to-Item Distance (I2I).** Item-to-item collaborative filtering [Sarwar *et al.*, 2001] utilises a database of ratings to compute item-to-item similarities, and predicts a current user's rating of an unseen item from his/her ratings of those items that are most similar to the item in question. Inspired by how item-to-item similarities are computed in this process, we use the observed log viewing times to derive the *I2I* distance measure as follows. We first transform the log viewing times into z-scores by normalising the values for each visitor separately. This ensures that varying viewing behaviour does not affect the similarity computation.[1] Secondly, we calculate item-to-item similarities using Pearson's correlation coefficient on the normalised log viewing times of exhibits i and j (using only the normalised log viewing times of those visitors that have viewed both exhibits i and j). The resulting similarity value from within the interval $[-1, 1]$ is finally transformed into a distance measure by mapping it onto a value in $[0, 1]$ (a similarity value of -1 yields a distance of 1, and a similarity of 1 yields a distance of 0).
- **Physical Distance (PD).** Museums are carefully themed by curatorial staff, such that closely-related exhibits are in physical proximity. Based on this observation, we hypothesise that physical walking distance between exhibits is inversely proportional to their (content) similarity. Thus, we use physical walking distance *PD* as a measure of distance between exhibits. Specifically, a SVG file-based representation of Melbourne Museum was used to calculate the walking distances by mapping the site map (Figure 1) onto a graph structure which preserves the physical layout of the museum (i. e., preventing paths from passing through walls or ceilings). We normalised the resulting distances to the interval $[0, 1]$.

4.3 Our Gaussian Spatial Process Model

In this section, we utilise theory from the area of spatial statistics (Section 4.2) to formulate a Gaussian spatial process model, called *Spatial Process Model (SPM)*, for predicting a museum visitor's interests in unseen exhibits (i. e., log viewing times) from his/her viewing behaviour at visited exhibits.

Let $U = \{1, \ldots, m\}$ be the set of all visitors, and $I = \{1, \ldots, n\}$ be the set of all items. Typically, for a visitor $u \in U$, we have viewing times for only a subset of I, say for n_u exhibits. Denoting a visitor's log viewing time vector with $\boldsymbol{r}_u$, we collect all observed log viewing times into a vector $\boldsymbol{r} = (\boldsymbol{r}_1, \ldots, \boldsymbol{r}_m)$ of dimension $\sum_{u=1}^{m} n_u$. Associated with each exhibit $i \in I$ is a log viewing time mean μ_i and a standard deviation σ_i. Let $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)$ be the vector of mean log viewing times, and $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n)$ the vector of standard deviations. Furthermore, $\boldsymbol{\mu}_u$ and $\boldsymbol{\sigma}_u$ are the vectors of means and standard deviations respectively for only those exhibits viewed by a visitor u. For example, if visitor 1 viewed exhibits 2, 3, 7 and 9, then $\boldsymbol{\mu}_1 = (\mu_2, \mu_3, \mu_7, \mu_9)$ and $\boldsymbol{\sigma}_1 = (\sigma_2, \sigma_3, \sigma_7, \sigma_9)$.

Similarly to spatial processes, *SPM* assumes a special correlation structure between the viewing times of different exhibits. In our experiments, we use a powered exponential [Banerjee *et al.*, 2004]:

$$\rho(\|\boldsymbol{s}_i - \boldsymbol{s}_j\|; \phi, \nu) = \exp\left(-\left(\phi\|\boldsymbol{s}_i - \boldsymbol{s}_j\|\right)^{\nu}\right),$$

where $\phi > 0$ and $0 < \nu < 2$. That is, $\rho(\|\boldsymbol{s}_i - \boldsymbol{s}_j\|; \phi, \nu)$ models the correlation between the log viewing times of exhibits i and j ($\rho(\|\boldsymbol{s}_i - \boldsymbol{s}_j\|; \phi, \nu)$ depends on the sites $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$ of exhibits i and j only through the distance $\|\boldsymbol{s}_i - \boldsymbol{s}_j\|$). Let $H(\phi, \nu)$ be a correlation matrix with components $(H(\phi, \nu))_{ij} = \rho(\|\boldsymbol{s}_i - \boldsymbol{s}_j\|; \phi, \nu)$ collecting all these correlations, and let $H_u(\phi, \nu)$ denote a visitor u's correlation matrix (dimension $n_u \times n_u$). That is, $H_u(\phi, \nu)$ corresponds to $H(\phi, \nu)$ without the rows and columns for unvisited exhibits. Also, let $\boldsymbol{\theta} = \left(\boldsymbol{\mu}, \boldsymbol{\sigma}, \tau^2, \phi, \nu\right)$ be a vector representing the $2n + 3$ model parameters, where τ^2 denotes the variance of non-spatial error terms necessary to fully specify the model (these terms model non-spatial variation in the data). Then, modelling the data using Gaussian spatial processes (a detailed derivation appears in [Bohnert *et al.*, 2009]), $\boldsymbol{r}$ given $\boldsymbol{\theta}$ is multivariate normal of dimension $\sum_{u=1}^{m} n_u$. As the viewing times of different visitors $u = 1, \ldots, m$ are independent, the model simplifies to

$$\boldsymbol{r}_u \,|\, \boldsymbol{\theta} \sim \mathcal{N}\left(\boldsymbol{\mu}_u, \Sigma_u\right) \text{ for all } u = 1, \ldots, m, \qquad (1)$$

where $\Sigma_u = \boldsymbol{\sigma}_u 1_{n_u} H_u(\phi, \nu) \boldsymbol{\sigma}_u 1_{n_u} + \tau^2 1_{n_u}$ is a visitor u's covariance matrix, and 1_{n_u} is the identity matrix of dimension $n_u \times n_u$.

We employ Bayesian inference using *SPM*'s likelihood function derived from Equation 1 to estimate $\boldsymbol{\theta}$ from $\boldsymbol{r}$ (in particular, we use *slice Gibbs sampling* [Neal, 2003]). This solution offers attractive advantages over the classic frequentist approach, such as the opportunity of incorporating prior knowledge into parameter estimation via the prior distribution, and capturing the uncertainty about the parameters via the posterior distribution.

Given the model parameters $\boldsymbol{\theta} = \left(\boldsymbol{\mu}, \boldsymbol{\sigma}, \tau^2, \phi, \nu\right)$, our model is fully specified, and we can use standard multivariate normal theory to predict a current visitor a's log viewing times of unseen exhibits, say $\boldsymbol{r}_{a,1}$, from a vector of observed log viewing times $\boldsymbol{r}_{a,2}$. This is because $(\boldsymbol{r}_{a,1}, \boldsymbol{r}_{a,2}) \,|\, \boldsymbol{\theta}$ is normally distributed (similarly to Equation 1). If we use the following notation

$$\begin{bmatrix} \boldsymbol{r}_{a,1} \\ \boldsymbol{r}_{a,2} \end{bmatrix} \,|\, \boldsymbol{\theta} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_{a,1} \\ \boldsymbol{\mu}_{a,2} \end{bmatrix}, \begin{bmatrix} \Sigma_{a,11} & \Sigma_{a,12} \\ \Sigma_{a,12}^T & \Sigma_{a,22} \end{bmatrix}\right),$$

then the conditional distribution $p\left(\boldsymbol{r}_{a,1} | \boldsymbol{r}_{a,2}, \boldsymbol{\theta}\right)$ is normal with mean vector and covariance matrix

$$\begin{aligned} \mathbb{E}\left(\boldsymbol{r}_{a,1} | \boldsymbol{r}_{a,2}, \boldsymbol{\theta}\right) &= \boldsymbol{\mu}_{a,1} + \Sigma_{a,12}\Sigma_{a,22}^{-1}\left(\boldsymbol{r}_{a,2} - \boldsymbol{\mu}_{a,2}\right), \\ \mathrm{Cov}\left(\boldsymbol{r}_{a,1} | \boldsymbol{r}_{a,2}, \boldsymbol{\theta}\right) &= \Sigma_{a,11} - \Sigma_{a,12}\Sigma_{a,22}^{-1}\Sigma_{a,12}^T, \end{aligned}$$

where $\mathbb{E}\left(\boldsymbol{r}_{a,1} | \boldsymbol{r}_{a,2}, \boldsymbol{\theta}\right)$ represents a personalised prediction of the log viewing times $\boldsymbol{r}_{a,1}$. Additionally, a measure of confidence in this prediction can be easily derived from

[1] We also tested a variant of the *I2I* measure without visitor-wise normalisation. However, this variant yielded inferior results.

$\mathrm{Cov}\,(\boldsymbol{r}_{a,1}|\boldsymbol{r}_{a,2},\boldsymbol{\theta})$, i. e., by using the variances on the diagonal of this matrix.

Being a model-based approach, *SPM* offers advantages over memory-based collaborative filters. For instance, the model parameters $\boldsymbol{\theta} = \left(\boldsymbol{\mu}, \boldsymbol{\sigma}, \tau^2, \phi, \nu\right)$ have a clear interpretation, and the confidence measure provided by the model supports an informed interpretation of the model's predictions. Additionally, recommendation generation is sped up by decoupling the model-fitting phase from the prediction phase.

5 Evaluation

This section reports on the results of an evaluation performed with our dataset (Section 3), including comparison with a nearest-neighbour collaborative filter.[2]

5.1 Experimental Setup

To evaluate the predictive performance of our *Spatial Process Model (SPM)*, we implemented two additional models: *Mean Model (MM)* and *Collaborative Filter (CF)*. *MM*, which we use as a baseline, predicts the $\log$ viewing time of an exhibit area i to be its (non-personalised) mean $\log$ viewing time μ_i. For *CF*, we implemented a nearest-neighbour collaborative filtering algorithm, and added modifications from the literature that improve its performance, such as shrinkage to the mean [James and Stein, 1961] and significance weighting [Herlocker *et al.*, 1999]. Additionally, to ensure that varying exhibit area complexity does not affect the similarity computation for selecting the nearest neighbours (viewing time increases with exhibit complexity), we transformed the $\log$ viewing times into z-scores by normalising the values for each of the exhibit areas separately. Visitor-to-visitor differences with respect to their mean viewing durations were removed by transforming predictions to the current visitor's viewing-time scale [Herlocker *et al.*, 1999]. Refer to [Bohnert and Zukerman, 2009] for a detailed description of *CF*. We tested several thousand different parameterisations, but in this paper, we report only on the performance of the best one.

Due to the relatively small dataset, we used leave-one-out cross validation to evaluate the performance of the different models. That is, for each visitor, we trained the models with a reduced dataset containing the data of 157 of the 158 visit trajectories, and used the withheld visitor pathway for testing. To train and instantiate the *SPM* variants (i. e., *SPM-I2I* and *SPM-PD*), we obtained a sample of $\boldsymbol{\theta} = \left(\boldsymbol{\mu}, \boldsymbol{\sigma}, \tau^2, \phi, \nu\right)$ from $p(\boldsymbol{\theta}|\boldsymbol{r})$ by performing slice Gibbs sampling [Neal, 2003] on the training data. For each of the 129 free model parameters,[3] we used (uninformative) independent uniform prior distributions. We used every 20-th sample after a burn-in phase of 1000 iterations as a sample of $\boldsymbol{\theta}$ from $p(\boldsymbol{\theta}|\boldsymbol{r})$, and stopped the sampling procedure after 8000 iterations. Thus, in total, we obtained 350 samples of $\boldsymbol{\theta}$ from $p(\boldsymbol{\theta}|\boldsymbol{r})$. This procedure was followed to obtain samples of $\boldsymbol{\theta}$ for both *SPM* variants, i. e., for both distance measures *I2I* and *PD* (Section 4.2). We then used the posterior means estimated from these samples to compute predictions by conditioning a multivariate normal distribution (Section 4.3). We improved *SPM-I2I*'s predictive performance by using the (non-personalised) mean $\log$ viewing time μ_i as a prediction whenever the conditioning would have been based on fewer than K $\log$ viewing times (in our case, $K = 19$). This modification was not applied to *SPM-PD*. For *CF*, predictions were computed from the ratings of the nearest neighbours; and for *MM*, we used μ_i, estimated from the appropriate reduced dataset, as a prediction.

We performed two types of experiments: *Individual Exhibit* and *Progressive Visit*.

- **Individual Exhibit (IE).** *IE* evaluates predictive performance for a single exhibit. For each observed visitor-exhibit area pair (u, i), we removed the observation r_{ui} from the vector of visitor u's $\log$ viewing durations, and computed a prediction $\hat{r}_{ui}$ from the other observations. This experiment is lenient in the sense that all available observations except the observation for exhibit area i are kept in a visitor's viewing duration vector.
- **Progressive Visit (PV).** *PV* evaluates performance as a museum visit progresses, i. e., as the number of viewed exhibit areas increases. For each visitor, we started with an empty visit, and iteratively added each viewed exhibit area to the visit history, together with its $\log$ viewing time. We then predicted the $\log$ viewing times of all yet unvisited exhibit areas.

For both experiments, we used the *mean absolute error (*MAE*)* to measure predictive accuracy as follows:

$$\mathrm{MAE} = \frac{1}{\sum_{u \in U} |I_u|} \sum_{u \in U} \sum_{i \in I_u} |r_{ui} - \hat{r}_{ui}|,$$

where I_u denotes a visitor u's set of exhibit areas for which predictions were computed. For *IE*, we calculated the total MAE for all valid visitor-exhibit area pairs; and for *PV*, we computed the MAE for the yet unvisited exhibit areas for all visitors at each time fraction of a visit (to account for different visit lengths, we normalised all visits to a length of 1).

5.2 Results

Table 2 shows the results for the *IE* experiment, where both spatial models (*SPM-I2I* and *SPM-PD*) outperform both *MM* and *CF*. Specifically, *SPM-I2I* achieves an MAE of 0.7756 (stderr 0.0068), and *SPM-PD* attains an MAE of 0.7548 (stderr 0.0066), outperforming *SPM-I2I* as well. The pairwise performance differences are statistically significant with $p \ll 0.01$ for all model pairings.

The performance of *SPM-PD*, *SPM-I2I*, *CF* and the baseline *MM* for the *PV* experiment is depicted in Figure 2. *CF* outperforms *MM* slightly (statistically significantly for visit fractions 0.191 to 0.374 and for several shorter intervals later on, $p < 0.05$). More importantly, both *SPM-I2I* and *SPM-PD* perform significantly better than *MM* and *CF*. For *SPM-I2I*, this performance increase is statistically significant for visit fractions 0.189 to 0.960 when comparing to *MM*, and except

[2] For our experiments, we ignore travel between exhibit areas, and collapse multiple viewing events of one area into one event.

[3] We set $\sigma_i = \sqrt{\sigma^2_{\boldsymbol{r},i} - \tau^2}$ to speed up the sampling process, where $\sigma^2_{\boldsymbol{r},i}$ denotes the sample variance of the $\log$ viewing times at exhibit i, calculated from the observed $\log$ viewing times r_{ui}. This reduces the number of free parameters from 255 ($126{\times}2{+}3$) to 129.

Table 2: Model performance for the *IE* experiment (MAE)

	MAE	Stderr
Mean Model (MM)	0.8618	0.0071
Collaborative Filter (CF)	0.7868	0.0068
Spatial Process Model using *I2I* (SPM-*I2I*)	0.7756	0.0068
Spatial Process Model using *PD* (SPM-*PD*)	0.7548	0.0066

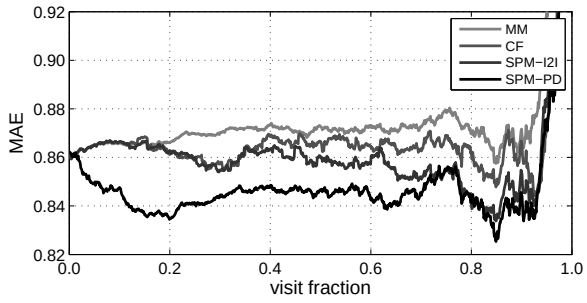


Figure 2: Model performance for the *PV* experiment (MAE)

for a few short intervals, for visit fractions 0.375 to 0.902 when comparing to *CF*. In comparison, *SPM-PD* performs significantly better than both *MM* and *CF* for visit fractions 0.019 to 0.922 (statistically significantly, $p < 0.05$). Additionally, *SPM-PD* outperforms *SPM-I2I* until visit fraction 0.660 (statistically significantly, $p < 0.05$). Drawing attention to the initial portion of the visits, *SPM-PD*'s MAE decreases rapidly, whereas the MAE for *MM* and *CF* remains at a higher level. Generally, the faster a model adapts to a visitor's interests, the more likely it is to quickly deliver (personally) useful recommendations. Such behaviour in the early stages of a museum visit is essential in order to build trust in the RS, and to guide a visitor in a phase of the visit where such guidance is most likely needed. A similar improvement in performance cannot be observed for *SPM-I2I*, which suggests that a visitor's exhibit interests observed in close physical proximity are better predictors of interests in unseen exhibits than interests in exhibits with positively correlated viewing times. As expected, *MM* performs at a relatively constant MAE level. For *CF*, *SPM-I2I* and *SPM-PD* we expected to see an improvement in performance (relative to *MM*) as the number of visited exhibit areas increases. However, this trend is rather subtle (it can be observed when plotting the models' performance relative to *MM*). Additionally, for all four models, there is a performance drop towards the end of a visit. We postulate that these phenomena may be explained, at least partially, by the increased influence of outliers on the MAE as the number of exhibit areas remaining to be viewed is reduced with the progression of a visit. This influence in turn offsets potential gains in performance obtained from additional observations. Our hypothesis is supported by a widening in the standard error bands for all models as a visit progresses, in particular towards the end (not shown in Figure 2 for clarity of presentation). However, this behaviour requires further, more rigorous investigation.

6 Discussion

SPM offers advantages over other model-based approaches in that, unlike neural networks (and memory-based techniques), it returns the confidence in a prediction, and its parameters have a clear interpretation; unlike Bayesian networks, our model does not require a domain-specific adaptation, such as designing the network topology. In addition, the distance measure endows our model with capabilities of hybrid RS [Burke, 2002; Albrecht and Zukerman, 2007] by seamlessly supporting the incorporation of other types of models (e. g., content-based). The distance measure also alleviates the *cold-start problem*. The *new-item problem* is addressed by utilising the (distance-based) correlation between this item and the other items. The *new-user problem* is similarly handled through the correlation between items rated by a user and the other items (when utilising *Physical Distance* as the distance measure, our model can make useful personalised predictions after only one item has been rated).

Our dataset is relatively small compared to other real-world RS applications. Although a high number of ratings per user slows down the slice Gibbs sampler due to repeated inversion of matrices of high dimension, employing our model with larger datasets should not represent a problem in practice. This is because the number of ratings per user is usually small compared to the number of users and items, and the computational complexity of evaluating the likelihood function depends only linearly on the number of users in the database.

7 Conclusions and Future Work

In this paper, we utilised the theory of spatial processes to develop a model-based approach for predicting users' interests in exhibits (i. e., implicit item ratings) within a RS in the museum domain. We applied our model to a real-world dataset collected by tracking visitors in a museum, using two measures of item-to-item (content) distance: (1) distances computed from item-to-item similarities (as in item-to-item collaborative filtering), and (2) physical walking distance. For both distance measures, our model attains a higher predictive accuracy than nearest-neighbour collaborative filters. Additionally, the model variant using physical distances outperforms that using distances computed from item-to-item similarities. Under the realistic *Progressive Visit* setting, our model using physical distance to measure item-to-item distance rapidly adapts to a user's ratings (starting from as little as one rating), thus alleviating the *new-user problem* common to collaborative filtering. This is not the case for the model variant based on distance computed from item-to-item similarities, which suggests that a visitor's interests observed for exhibits in close physical proximity are better predictors of interests in unseen exhibits than those interests for exhibits with positively correlated viewing times.

In the future, we intend to hybridise our model by incorporating content-based item features into our distance measure, and to explore hybrids of models utilising a variety of item-to-item distances. We also plan to extend our model to fit non-Gaussian item ratings, e. g., [Diggle *et al.*, 1998; Yu *et al.*, 2006].

Acknowledgments

This research was supported in part by grant DP0770931 from the Australian Research Council. The authors thank Carolyn Meehan and her team from Museum Victoria for their assistance; and David Abramson, Jeff Tan and Blair Bethwaite for their help with the computer cluster.

References

[Albrecht and Zukerman, 2007] D.W. Albrecht and I. Zukerman. Introduction to the special issue on statistical and probabilistic methods for user modeling. *User Modeling and User-Adapted Interaction*, 17(1-2):1–4, 2007.

[Aroyo *et al.*, 2007] L. Aroyo, N. Stash, Y. Wang, P. Gorgels, and L. Rutledge. CHIP demonstrator: Semantics-driven recommendations and museum tour generation. In *Proceedings of the Sixth International Conference on the Semantic Web (ISWC-07)*, pages 879–886, 2007.

[Banerjee *et al.*, 2004] S. Banerjee, B.P. Carlin, and A.E. Gelfand. *Hierarchical Modeling and Analysis for Spatial Data*. Chapman & Hall/CRC, 2004.

[Bell *et al.*, 2007] R. Bell, Y. Koren, and C. Volinsky. Chasing $1,000,000: How we won the Netflix progress prize. *ASA Statistical and Computing Graphics Newsletter*, 18(2):4–12, 2007.

[Bohnert and Zukerman, 2009] F. Bohnert and I. Zukerman. Non-intrusive personalisation of the museum experience. In *Proceedings of the 1st and 17th International Conference on User Modeling, Adaptation, and Personalization (UMAP-09)*, pages 197–209, 2009.

[Bohnert *et al.*, 2008] F. Bohnert, I. Zukerman, S. Berkovsky, T. Baldwin, and L. Sonenberg. Using interest and transition models to predict visitor locations in museums. *AI Communications*, 21(2-3):195–202, 2008.

[Bohnert *et al.*, 2009] F. Bohnert, D.F. Schmidt, and I. Zukerman. Spatial processes for recommender systems. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI-09)*, 2009.

[Breese *et al.*, 1998] J.S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th International Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 42–52, 1998.

[Burke, 2002] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

[Cheverst *et al.*, 2002] K. Cheverst, K. Mitchell, and N. Davies. The role of adaptive hypermedia in a context-aware tourist guide. *Communications of the ACM*, 45(5):47–51, 2002.

[Diggle *et al.*, 1998] P.J. Diggle, J.A. Tawn, and R.A. Moyeed. Model-based geostatistics. *Applied Statistics*, 47(3):299–350, 1998.

[Hatala and Wakkary, 2005] M. Hatala and R. Wakkary. Ontology-based user modeling in an augmented audio reality system for museums. *User Modeling and User-Adapted Interaction*, 15(3-4):339–380, 2005.

[Herlocker *et al.*, 1999] J.L. Herlocker, J.A. Konstan, A. Borchers, and J.T. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-99)*, pages 230–237, 1999.

[James and Stein, 1961] W. James and C.M. Stein. Estimation with quadratic loss. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1*, pages 361–379, 1961.

[Neal, 2003] R.M. Neal. Slice sampling. *The Annals of Statistics*, 31(3):705–767, 2003.

[Parsons *et al.*, 2004] J. Parsons, P. Ralph, and K. Gallager. Using viewing time to infer user preference in recommender systems. In *Proceedings of the AAAI Workshop on Semantic Web Personalization (SWP-04)*, pages 52–64, 2004.

[Petrelli and Not, 2005] D. Petrelli and E. Not. User-centred design of flexible hypermedia for a mobile guide: Reflections on the HyperAudio experience. *User Modeling and User-Adapted Interaction*, 15(3-4):303–338, 2005.

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

[Sarwar *et al.*, 2001] B. Sarwar, G. Karypis, J.A. Konstan, and J.T. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on the World Wide Web (WWW-01)*, pages 285–295, 2001.

[Schmidt *et al.*, 2009] D.F. Schmidt, I. Zukerman, and D.W. Albrecht. Assessing the impact of measurement uncertainty on user models in spatial domains. In *Proceedings of the 1st and 17th International Conference on User Modeling, Adaptation, and Personalization (UMAP-09)*, pages 210–222, 2009.

[Schwaighofer *et al.*, 2005] A. Schwaighofer, V. Tresp, and K. Yu. Learning Gaussian process kernels via hierarchical Bayes. In *Advances in Neural Information Processing Systems 17 (NIPS-04)*, pages 1209–1216, 2005.

[Schwarz, 1978] G.E. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[Stock *et al.*, 2007] O. Stock, M. Zancanaro, P. Busetta, C. Callaway, A. Krüger, M. Kruppa, T. Kuflik, E. Not, and C. Rocchi. Adaptive, intelligent presentation of information for the museum visitor in PEACH. *User Modeling and User-Adapted Interaction*, 18(3):257–304, 2007.

[Yu *et al.*, 2006] S. Yu, K. Yu, V. Tresp, and H.-P. Kriegel. Collaborative ordinal regression. In *Proceedings of the 23rd International Conference on Machine Learning (ICML-06)*, pages 1089–1096, 2006.

Optimal Set Recommendations based on Regret

Paolo Viappiani
Department of Computer Science
University of Toronto
Toronto, ON, Canada
paolo@cs.toronto.edu

Craig Boutilier
Department of Computer Science
University of Toronto
Toronto, ON, Canada
cebly@cs.toronto.edu

Abstract

Current conversational recommender systems do not offer guarantees on the quality of their recommendations, either because they do not maintain a model of a user's utility function, or do so in an *ad hoc* fashion. In this paper, we propose an approach to recommender systems that incorporates explicit utility models into the recommendation process in a decision-theoretically sound fashion. The system maintains explicit constraints on the user's utility based on the semantics of the preferences revealed by the user's actions. In particular, we propose and investigate a new decision criterion, *setwise maximum regret*, for constructing optimal recommendation sets. This new criterion extends the mathematical notion of *maximum regret* used in decision theory and preference elicitation to sets. We develop computational procedures for computing setwise max regret. We also show that the criterion suggests choice sets for queries that are myopically optimal: that is, it refines knowledge of a user's utility function in a way that reduces max regret more quickly than any other choice set. Thus setwise max regret acts both as guarantee on the quality of our recommendations and as a driver for further utility elicitation.

Our simulation results suggest that this utility-theoretically sound approach to user modeling allows much more effective navigation of a product space than traditional approaches based on, for example, heuristic utility models and product similarity measures.

Introduction

Recommender systems can help users navigate product spaces and make decisions involving very large sets of alternatives. *Conversational* recommender systems rely on mixed-initiative interactions, with both the user and the system taking an active role in the decision process. User feedback can be entered in many forms, for instance, as direct answers to queries, or *critique* of the options displayed by the system.

Many recommender systems employ some form of *diversity* to show a set of products that might be appealing to the user. Intuitively, diversity overcomes a key problem with presentation of the *top-k* items based on some estimate of a user's score: the latter tends to produce results that are very similar one to each other, and thus not offer much actual "choice" for a user. This is especially true when we recognize that estimated scores or preferences are likely to be very crude. Diversity is also important in practice: we cannot generally predict how *patient* a user will be. They may terminate the exploration of product space at any time, hence the recommender system should be able to provide *anytime* recommendations, reflecting the best recommendations given the information provided by the user so far. This characteristic of conversational recommenders is similar to the exploration-exploitation dilemma in reinforcement learning. Since we do not know know how much time the user is willing to spend in order to improve the recommendation, we want to show products that are both: (a) expected to be rated highly given the current information about the user; and (b) are maximally informative should the user critique (or otherwise provide feedback on) them.

Many authors have considered the importance of diversity in the recommendations. For example, researchers in case-based reasoning have proposed techniques based on greedy maximization of diversity (McSherry 2002), defined as an aggregate of a distance metric, or as a weighted tradeoff between diversity and the recommendation score (Smyth and McClave 2001). However diversity and dissimilarity measures does not consider the information that we have about the user's preference. While they guarantee that the set contains alternatives that differ in their features, they do not use at all the information about a user's preferences available from previous user actions and feedback. It has been argued that diversity should be instead *tailored* to the system's belief about the user (Price and Messinger 2005).[1]

To maximize the information presented to the user in a recommendation set, and recommend a set of *optimal* recommendations, it is necessary to maintain an explicit representation of the uncertainty in the preference model and a sound decision-theoretic semantics of the interaction in the first place. In fact, most practical conversational recommender systems (especially those using critiquing) do not use an explicit model of a user's preferences, or only main-

[1] Indeed, the natural decision theoretic account of set recommendations immediately suggests diversity w.r.t. belief about a user's preferences (Boutilier *et al.* 2003).

tain such a model in an ad hoc, heuristic fashion. In this paper, we develop an approach to set-based recommendations with an explicit utility model. We represent the uncertainty w.r.t. the user model with constraints of her utility function induced by choices or critiques. To construct a suitable recommendation set, we develop a novel criterion, *setwise maximum regret*, that captures the idea of providing a set of *jointly* optimal recommendations. Our qualitative model of uncertainty has two key advantages over probabilistic models (Price and Messinger 2005): relatively simple prior information in the form of bounds or constraints on user preferences can be exploited (rather than probabilistic priors); and exact computation is much more tractable (in contrast with probabilistic models of utility that generally require reasoning with densities that have no closed form (Boutilier 2002; Chajewska and Koller 2000)).

To make this model effective, user actions should be associated with a precise, sound *semantics*. For instance, a user critique is assumed to reveal some aspect of the user's preferences and this is used to update an explicit utility model. More precisely, in our work, unit critiques and compound critiques places linear constraints on a user utility function. The advantage of this approach is that we can use decision-theoretically sound criteria to:

1. suggest or recommend a product;
2. bound the difference in the quality of a recommended product and the optimal option for the user;
3. determine which options and critiques carry the most information to help speed up the navigation process; and
4. suggest to the user when to terminate the process (i.e., when further interaction will offer only modest improvement in recommendation quality).

We adopt the notion of *minimax regret* (Boutilier *et al.* 2006a) to make product suggestions in the face of utility function uncertainty. This robust decision criterion allows us to bound the loss (difference from optimal) of any recommendation. We propose and investigate a new decision criterion, *setwise maximum regret*, for constructing optimal recommendation sets. This new criterion extends maximum regret to sets of products rather than a single product. We define set maximum regret, argue that minimizing setwise max regret is the best means for constructing a set of options for a user, and develop effective computational procedures for computing optimal recommendation sets for setwise regret.

We present *critiquing* as a possible application domain. While user-controlled exploration in traditional critiquing systems does not offer any guarantees (practical, empirical, or theoretical) of either sufficient or efficient exploration of the space (A user may cycle through a set of similar products or converge at a product far from optimal), our regret-based recommender allows us to provide guarantees on the quality (utility) of the recommended product vis-à-vis feasible alternatives. We also show with simulations that *regret-based critiquing* can lead to much more efficient exploration of the product space and lead to better decisions in practice.

In Sec. 2 we introduce our model of regret-based recommendation and describe our strategy for selection of a joint set of recommended alternatives using setwise minimax regret. In Sec. 3 we discuss computation of setwise max regret and minimax regret, both for configuration problems modeled as a constraint satisfaction problem (CSP) and for product databases, while in Sec. 4 we briefly discuss the performance of elicitation. Finally, in Sec. 5, we perform simulations of complete critiquing-based recommender systems, comparing our regret-based approach to state of the art critiquing algorithms such as dynamic critiquing and incremental critiquing.

Regret-based Recommendation Systems

We begin this section by presenting our formalization of the decision problem, reviewing minimax regret for robust recommendation and elicitation, and then defining our key concepts of setwise max regret and setwise minimax regret.

Underlying Decision Problem

We assume a recommendation system is charged with the task of recommending an option to a user in a multi attribute space (e.g., computers, cars, apartment rental, etc.). Products are characterized by a finite set of attributes $\mathcal{X} = \{X_1, ...X_n\}$, each with finite domains $Dom(X_i)$. Let $\mathbf{X} \subseteq Dom(\mathcal{X})$ denote the set of *feasible configurations*. For instance, attributes may correspond to the features of various apartments, such as size, neighborhood, distance from public transportation, etc., with $\mathbf{X}$ defined either by constraints on attribute combinations (e.g., constraints on computer components that can be put together), or by an explicit database of feasible configurations (e.g., a rental database).

The user has a *utility function* $u : Dom(\mathcal{X}) \rightarrow \mathbf{R}$. In what follows we will assume either a *linear* or *additive* utility function depending on the nature of the attributes (Keeney and Raiffa 1976). In both additive and linear models, we assume that u can be decomposed as follows:

$$u(\mathbf{x}) = \sum_i f_i(x_i) = \sum_i \lambda_i v_i(x_i)$$

where each local utility function f_i assigns a value to each element of $Dom(X_i)$. In classical utility elicitation, these values can be determined by assessing local value functions v_i over $Dom(X_i)$ that are normalized on the interval $[0, 1]$, and importance weights λ_i ($\sum_i \lambda_i = 1$) for each attribute (Keeney and Raiffa 1976; Fishburn 1967). This sets $f_i(x_i) = \lambda_i v_i(x_i)$ and ensures that global utility is normalized on the interval $[0, 1]$. A simple additive model in the rental domain might be:

$$u(Apt) = f_1(Size) + f_2(Distance) + f_3(Nbrhd)$$

When $Dom(X_i)$ is drawn from some real-valued set, we often assume that v_i (hence f_i) is linear in X_i.[2]

Since a user's utility function is not generally known, we often write $u(\mathbf{x}; w)$ to emphasize the dependence of u on

[2]Our approach relies considerably on the additive assumption, though can easily be generalized to more general models such as GAI (Fishburn 1967; Bacchus and Grove 1995; Braziunas and Boutilier 2007a). The assumption of linearity is simply a convenience; nothing critical depends on it.

user-specific parameters. In the additive case, the values $f_i(x_i)$ over $\cup_i\{Dom(X_i)\}$ serve as a sufficient parametrization of u (for linear attributes, a more succinct representation is possible). The optimal product for the user with utility parameters w is that $\mathbf{x} \in \mathbf{X}$ that maximizes $u(\mathbf{x}; w)$. Our goal is to recommend, or help the user find, an optimal product, or one whose utility is near optimal.

Regret-based Recommendation

In probabilistic approaches to recommendation, a distribution over preferences—typically in the form a density over utility function parameters—is maintained, and the option with highest expected utility is recommended (Chajewska *et al.* 2000; Boutilier 2002; Boutilier *et al.* 2003). When a set of alternatives need to be recommended, the expectimax or EMAX criterion can be used (Boutilier *et al.* 2003; Price and Messinger 2005). One difficulty with probabilistic models is that one requires probabilistic prior information over utility models, which can be difficult to formulate and represent. Another is that exact computation can often be computationally intense; this is especially true since (arguably) natural density models for utility functions are rarely closed under the type of evidence provided by user interaction (e.g., behavioral observation or answers to queries) (Boutilier 2002; Chajewska and Koller 2000)); as a result, computationally demanding fitting of (say) mixture models is required after every model update.

Instead, we propose the use of minimax regret to generate *recommendation sets*. As we will see, this obviates the need to complex probabilistic reasoning, yet can offer robust recommendations and provide very effective guidance for the user. In traditional regret-based approaches, a single recommendation is made using the minimax regret (Savage 1954) criterion. For multiple joint recommendations, we develop the notion of *setwise minimax regret* (defined below). We can summarize the correspondence between the Bayesian and the regret-based approach with the following table:

Probabilistic approach	Regret approach
Expected Utility	Minimax Regret
Expected Max (EMAX)	Minimax *Setwise* Regret

In this paper we propose a framework that maintains a set W of feasible utility models, and at each step, the system shows a set of recommendations that are *jointly* optimal with respect to minimax regret. At a very high level, our regret-based recommender works as follows:

1. The set W is initialized given some initial constraints;
2. The current recommendations are determined (using the setwise minimax regret);
3. After each user action, W is refined to reflect the new constraints imposed by the user's feedback;
4. The process repeats (steps 2 and 3) until the user is satisfied or minimax regret reaches some target threshold.

This process is appealing for two reasons. First, the current recommendation (i.e., set of options) is always optimal; in other words, it minimizes setwise max regret given the current information about the user's utility function, making it extremely robust in the presence of utility function uncertainty (in a way to be made precise below). Second, max regret is a well-defined progress metric that lets the user know the cost and benefit of further exploration of product space. Finally, the information contained in user selection of some choice from the recommended set is maximally informative (in a sense defined below).

Minimax Regret

Minimax regret has been advocated as a means for robust optimization (Kouvelis and Yu 1997), and has more recently been used for decision making with utility uncertainty (Boutilier *et al.* 2001; Salo and Hämäläinen 2001; Boutilier *et al.* 2006a).

Assume that through some interactions with a user, and possibly using some prior knowledge, we determine that her utility function w lies in some set W. Following (Boutilier *et al.* 2006a) we define:

Definition 1 *Given a set of feasible utility functions W, we define the* pairwise max regret $MR(\mathbf{x}, \mathbf{y}; W)$ *of* $\mathbf{x}, \mathbf{y} \in \mathbf{X}$*; the* max regret $MR(\mathbf{x}; W)$ *of* $\mathbf{x} \in \mathbf{X}$*; the* minimax regret $MMR(W)$ *of W; and the* minimax optimal configuration $\mathbf{x}^*_W$ *as follows:*

$$MR(\mathbf{x}, \mathbf{y}; W) = \max_{w \in W} u(\mathbf{y}; w) - u(\mathbf{x}; w) \quad (1)$$

$$MR(\mathbf{x}; W) = \max_{\mathbf{y} \in \mathbf{X}} MR(\mathbf{x}, \mathbf{y}; W) \quad (2)$$

$$MMR(W) = \min_{\mathbf{x} \in \mathbf{X}} MR(\mathbf{x}, W) \quad (3)$$

$$\mathbf{x}^*_W = \arg\min_{\mathbf{x} \in \mathbf{X}} MR(\mathbf{x}, W) \quad (4)$$

Intuitively, $MR(\mathbf{x}; W)$ is the worst-case loss associated with recommending configuration $\mathbf{x}$; i.e., by assuming an adversary will choose the user's utility function w from W to maximize the difference in utility between the optimal configuration (under w) and $\mathbf{x}$. The minimax optimal configuration $\mathbf{x}^*_W$ minimizes this potential loss. $MR(\mathbf{x}, W)$ bounds the loss associated with $\mathbf{x}$, and is zero iff $\mathbf{x}$ is optimal for all $w \in W$. Any choice that is not minimax optimal has strictly greater loss than $\mathbf{x}^*_W$ for some $w \in W$.

Minimax regret has proven to be an effective tool in utility elicitation in a variety of domains. A decision support or recommender system can query (or otherwise interact with) a user providing additional constraints on the utility set W until minimax regret reaches some acceptable level (possibly optimality), elicitation costs become too high, or some other termination criterion is met.

Example Consider the following example, where the options o^i are defined using two features/coordinates x_1 and x_2:

	x_1	x_2
o^1	0.35	0.68
o^2	0.9	0.2
o^3	0	0.75
o^4	1	0
o^5	0.5	0.3

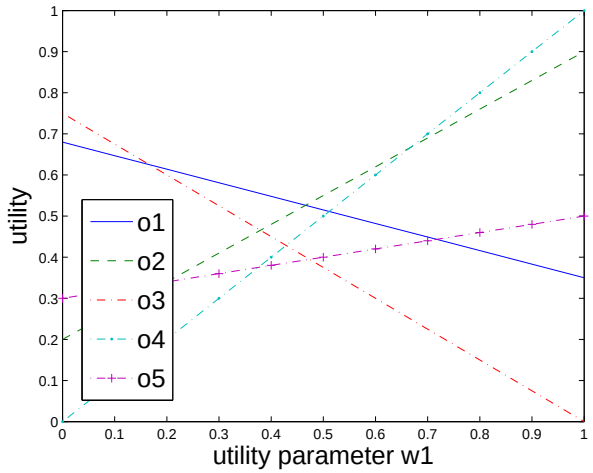


Figure 1: Each options is represented by a single line in the utility space.

We assume linear utility: $u(\mathbf{x};\mathbf{w}) = w_1x_1 + w_2x_2$ where $\mathbf{w}$ is vector of tradeoff weights, with $w_2 = 1-w_1, 0 \leq w_1 \leq 1$; and the local value functions for each coordinate are identity functions. (i.e., $v_i(x_i) = x_i$). Given these assumptions, utility is one-dimensional; it can be written as $u(\mathbf{x};\mathbf{w}) = (x_1 - x_2)w_1 + x_2$. So we deal only with the uncertainty on the single parameter w_1.

This simple example is convenient because it is easy to visualize option utilities as a 1D function of w_1 graphically. The utility of the different options are shown in Fig. 1 with respect to the parameter w_1. We notice that, for some values of w_1, each of the options o_1, o_2, o_3 and o_4 is optimal, but not so o_5. When considering a particular value of w_1 (a particular utility function) the *actual regret* (or real loss) is the difference between the utility of the best option given w_1 and the utility of our recommendation in that case. For instance, for $w_1 = 0.9$, the best option is o_4 with utility 0.9, while o_1 has utility 0.38, so the actual regret of o_1 would be $0.9 - 0.38 = 0.52$. Max regret accounts for the uncertainty over w_1 and it is the maximum of the possible actual regret values (for o_1 is 0.65 when $w_1 = 1$).

When the options are few as in this case, we can compute the max regret of each choice by explicitly enumerating the maximum the pairwise regret of that choice against any possible adversarial choice of option. The table below illustrates this, where each row corresponds to a recommendation, each column to an adversarial choice, and we display the pairwise max regret (allowing the adversary to choose utility) in the cells. The max regret of an option is shown in the last column, and corresponds to the maximum value in its row. In the unconstrained situation (where w_1 can take any value between 0 and 1), we have the following values:

$MR(o^i, o^j)$	o^1	o^2	o^3	o^4	o^5	$MR(o_i)$
o^1	0	0.55	0.07	0.65	0.15	0.65
o^2	0.48	0	0.55	0.1	0.1	0.55
o^3	0.35	0.9	0	1.00	0.5	1
o^4	0.68	0.2	0.75	0	0.3	0.75
o^5	0.38	0.4	0.45	0.5	0	0.5

Minimax regret is 0.5, and the minimax optimal recommendation is option o_5; its max regret occurs at adversarial choice of utility $w_1 = 1$, and choice of option o_4. It can be easily shown that regret is maximized at one of the vertexes of the feasible region W when W is a bounded, convex polytope (such a polytope induced by the interactions we discuss later).

Now imagine that, perhaps as the result of interactions with the user, we learn that $0.2 \leq w_1 \leq 0.6$. The new minimax regret value for this constrained case is 0.138. This value corresponds to recommendation o_1, and adversarial option o_2 and utility $w_1 = 0.6$. In this case, the constraints $0.2 \leq w_1$ and $w_1 \leq 0.6$ decrease minimax regret significantly. However usually constraints are not added directly as such, but result from the acquisition of knowledge acquired through a variety of interaction modalities, such as direct user preference queries, or passive observation of user behavior. For instance, comparison queries ask the user which of two proposed options is preferred. The impact of the information acquired depends greatly on the comparison, as different options can lead to different degrees of regret reduction.

A natural meta-heuristic for generating elicitation queries is the *current solution strategy (CSS)*, first described in (Boutilier *et al.* 2006a). This strategy would ask the user whether she prefers the minimax regret option x^*_W or the adversary option $x^w = MRAdv(x^*_W, W)$.

In our example, starting from the unconstrained space W, CSS would select $\{o_4, o_5\}$ (the minimax regret option and the adversarial option) and ask the user to compare them. Now, let's assume that the user asserts that she prefers o_4 over o_5; then a new constraint $u(o_4; w_1) \geq u(o_5; w_1)$, equivalent to $w_1 \geq 0.375$, is added to our model. In the space $W^{o_4 \succ o_5}$ resulting from the incorporation of this constraint, the minimax regret is 0.1, resulting from recommendation o_2 and adversarial option o_4. Option o_4, even if known to be better than o_5, has max regret of 0.18 (at $w_1 = 0.374$, with adversarial option o_1). Therefore, option o_2 will be recommended.

Minimax regret offers recommendations that are robust given the uncertainty of the preference model. In this example, o_5 is recommended (in the unconstrained setting) even though it cannot possibly be optimal for *any* user utility function; this is so because it prevents "disastrous" situations, such as would occur if options o_1 or o_2 are recommended when w_1 is very low (despite the fact that for a good part of utility space, these options are optimal. Note, that as knowledge of user utility increases, more accurate recommendations are made; for example, recommending o_2 when we learn that o_4 is preferred to o_5.

Optimal Recommendation Sets: Setwise Regret

In most cases the value of a set of recommendations is dependent on the elements of the set jointly, not on each individually. If the user is going to benefit from only one of the recommendations (example: recommending apartments) then the utility of the set is then the maximum utility among the individual options, i.e., the one the user will pick from the set.

The problem of set recommendations has been addressed using probabilistic expectation: Price and Messinger (Price and Messinger 2005) optimize set recommendations using the EMAX criterion, defined as the expectation of the maximum utility among the options in the set.

In order to retrieve optimal set recommendations, we define the notion of *setwise max regret*. The setwise max regret of a recommendations set can be seen as the equivalent of EMAX in our non-probabilistic framework. Suppose we have a slate of k options to present to the user and want to quantify the possible loss by restricting the user's decision to options in that slate. Intuitively, the user may select any of the k options as being "optimal." An adversary wanting to maximize regret should do so assuming the any such choice is possible—unlike max regret, we allow the user to select from among any of the set of k options. In this formalization, we choose the set of k options first, but delay the final choice from the slate only *after* the adversary has chosen a utility function w. The regret of a set is then the minimum difference between the utility of the best configuration under w and the utility of the options in the slate. Specifically, define the *setwise maximum regret* of option set $\mathbf{Z} = \{\mathbf{x}^1, \ldots, \mathbf{x}^j\}$ to be:

$$SMR(\mathbf{Z}; W) = \max_{\mathbf{x}' \in \mathbf{X}} \max_{w \in W} \min_{\mathbf{x} \in \mathbf{Z}} u(\mathbf{x}'; w) - u(\mathbf{x}; w)$$

$$SMR\text{-}Adv(\mathbf{Z}; W) = \arg\max_{\mathbf{x}' \in \mathbf{X}} \max_{w \in W} \min_{\mathbf{x} \in \mathbf{Z}} u(\mathbf{x}'; w) - u(\mathbf{x}; w)$$

Setwise max regret has some intuitive properties. First, adding new items to a set cannot increase setwise max regret: $SMR(\mathbf{A} \cup \mathbf{B}, W) \leq SMR(\mathbf{A}, W)$. At the same time incorporating options that are known to be dominated given W does not change setwise max regret: in other words, if $u(\mathbf{a}, \mathbf{w}) > u(\mathbf{b}, \mathbf{w})$ for some $\mathbf{a} \in \mathbf{Z}$ and all $\mathbf{w} \in W$, then $SMR(\mathbf{Z} \cup \{\mathbf{b}\}, W) = SMR(\mathbf{Z}, W)$. Finally, the max regret associated with recommending the entire product set is zero: $SMR(\mathbf{X}, W) = 0$. This is the equivalent to asking the user to directly choose the best option from the space of available options—obviously, a task of with extreme cognitive cost, and one that runs counter to the spirit of recommendation assistance! But should the user be able to answer correctly, it guarantees optimality.

Setwise max regret can be equivalently written in as follows:

$$SMR(\mathbf{Z}, W) = \max_{\mathbf{y} \in \mathbf{X}} \max_{w \in W} [u(\mathbf{y}; w) - \max_{\mathbf{x} \in \mathbf{Z}} u(\mathbf{x}; w)] \quad (5)$$

This captures the intuition that, given w, the option (among those in $\mathbf{Z}$) that determines setwise max regret is that with highest utility with respect to w. In fact, it can be useful to explicitly partition utility space with respect to which option in $\mathbf{Z}$ is maximal. We define the utility subset $W^{\mathbf{Z} \to \mathbf{x}_i}$ as the set of utilities such that $\mathbf{x}_i$ has greater utility than any option in $\mathbf{Z}$.

$$W^{\mathbf{Z} \to \mathbf{x}_i} = \{w \in W : u(\mathbf{x}_i; w) > u(\mathbf{x}_j; w)\ \forall j \neq i, 1 \leq j \leq k\}$$

The set of all $W^{\mathbf{Z} \to \mathbf{x}_i}$ for any $\mathbf{x}_i \in \mathbf{Z}$ partitions W (we ignore the possibility of ties over full-dimensional subsets of W, which can easily be dealt with, but complicate the presentation marginally). An important observation (that will be used later) is that we can rewrite the setwise max-regret SMR as the aggregate maximum of the (individual) max-regret considering a partition of the utility space according to which option has higher utility.

Observation 1 *Given* $\mathbf{Z} = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$ *and, for* $1 \leq i \leq k$*,*

$$SMR(\mathbf{Z}, W) = max[MR(\mathbf{x}_1, W^{Z \to \mathbf{x}_1}), \ldots, MR(\mathbf{x}_k, W^{Z \to \mathbf{x}_k})] \quad (6)$$

Example (continued) We now consider setwise max regret for the example introduced above. Let the number of options in a recommendation set be $k = 2$. The following combinations are ranked best according to the setwise regret criterion.

Set	SMR	Adversary	Adversary W
$\{o_1, o_4\}$	0.07	o_3	$w_1 = 0$
$\{o_1, o_2\}$	0.1	o_4	$w_1 = 1$
$\{o_3, o_2\}$	0.1	o_4	$w_1 = 1$
$\{o_3, o_4\}$	0.11	o_1	$w_1 = 0.42$

The set $\{o_1, o_4\}$ is the best choice for a joint recommendation of two options, corresponding to a value of regret of 0.07. Other combinations, such as $\{o_1, o_2\}$, $\{o_3, o_2\}$ and $\{o_3, o_4\}$, also have a relative low value of regret.

A set recommendation can often have dramatically lower regret than the minimax optimal *single* recommendation (in this case, o_5).

It is interesting the fact that the optimal recommendation set is composed of two options, o_1 and o_4 that, when considered alone, are associated with high regret. Any set including o_5, the single best recommendation, is ranked poorly with respect to setwise regret.

We now consider the case of larger sets. If we need to select a slate of three options ($k = 3$), the regret will be 0.04 and the recommendation would be $\{o_1, o_3, o_4\}$; in this case the adversary would pick o_2, and the value $w_1 = 0.51$ (intersection point of o_1 and o_4).

In the case of four options to be selected ($k = 4$), the set $\{o_1, o_2, o_3, o_4\}$ would be recommended and it would be associated to a setwise regret of 0: the slate includes all the options that can ever become optimal (considering Observation 1, it follows that for any $W^{\mathbf{Z} \to o_i}$ that partitions W, the max regret has to be 0).

Now we consider how setwise regret changes when new information is included. We consider a slate of two options to be selected ($k = 2$) and we suppose that the user asserts the preference of o_4 over o_5. The recommendation set is still

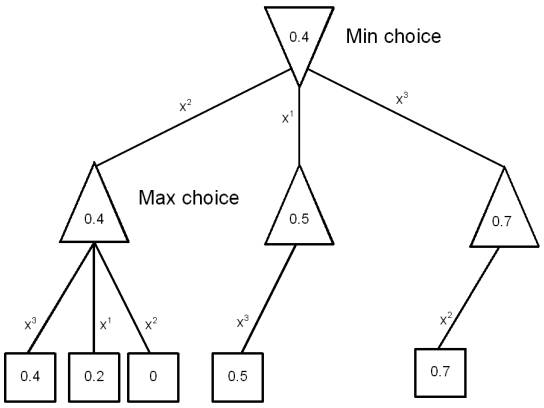


Figure 2: Alpha beta pruning can speed up the search, depending on the evaluation order. In this case, x^1 has regret 0.5 against x^3, that is worse than the value 0.4 (max regret of x^2), so we do not need to test x^1 against x^2.

$\{o_1, o_4\}$ but with a much lower value of (setwise) regret: only 0.04.

We conclude the discussion of the example with some remarks on the optimization process. The adversary's utility does not necessarily corresponds to one the vertex of the feasible region, as in the single recommendation case; it may also lie in any intersection of the hyperplanes associated with the options. For instance (in the unconstrained case) the setwise regret of $\{o_3, o_4\}$ is maximized for the value $w_1 = 0.42$ (the utility that makes o_3 and o_4 equally preferred).

Computation of Setwise Minimax Regret

In this section we discuss how to efficiently compute regret-based recommendations. We first discuss how to compute minimax regret for single recommendations and then describe how to modify these procedures to compute setwise minimax regret for recommendation sets. We distinguish two settings: *configuration problems*, where options are defined by variables and configuration constraints (i.e., as solutions to a constraint satisfaction problem (CSP)); and *database problems*, where options are enumerated in a product database.

Computing Minimax Optimal Single Recommendations

Configuration problems In configuration problems, optimization over product space $\mathbf{X}$ is formulated as a constraint optimization problem or MIP. In such domains, minimax regret computation can be formulated as a MIP, and solved practically for large problems using techniques such as Bender's decomposition and constraint generation. We refer to (Boutilier *et al.* 2006a; 2004; Braziunas and Boutilier 2007a) for more details. Our MIP formulations for setwise minimax regret below will draw heavily on these techniques, but necessitate important modifications.

Database problems When options are enumerated in a product database, minimax regret computation requires to repeated computation of the pairwise regret between a candidate recommendation and an adversarial option in order to identify the option with minimax regret. For ease of presentation, assume a linear utility function as above, defined by weights w_i over m attributes. Pairwise regret $MR(\mathbf{x}, \mathbf{y}, W)$ of recommendation $\mathbf{x}$ and adversarial option $\mathbf{y}$ is readily computed with the following LP:

$$\max_{\mathbf{w}: w_i \in [0,1]} \sum_{1 \leq i \leq m} w_i(y_i - x_i) \quad (7)$$

$$\text{s.t.} \sum_i w_i = 1 \quad (8)$$

$$\mathbf{w} \in W \quad (9)$$

Here we assume the feasible parameter set W is captured by linear constraints. A similar LP can be formulated for discrete-valued attributes, without assuming linearity, just additivity. Hybrid models with continuous and discrete attributes can easily be represented with a combination of these two representations. Generalized additive utility models models (Fishburn 1967; Braziunas and Boutilier 2006; 2007b) can also be easily represented in this framework. This means that pairwise regret can be computed extremely efficiently (e.g., in a few milliseconds using CPLEX on the types of problems discussed below).

Minimax regret computation is more complex because we need to maximize over all possible adversarial choices, and minimize over all possible recommendations. A naive approach would consider every pair of options, requiring $O(n^2)$ pairwise regret computations for a database of size n, where each of these computations requires the solution of an LP of size proportional to the number of utility parameters.

However, since minimax regret can be seen as a game between the recommender and an adversary, the computation can be greatly improved in practice by formulating the optimization as a minimax search and using standard pruning techniques. Unlike typical games, the search tree has very limited depth: only two ply, one choice of recommendation by the MIN player (attempting to minimize regret) and one choice of adversarial option by the MAX player (attempting to maximize the regret of the recommendation).[3] Note that the game has a large number of actions, once per product in the database. The MIN player (recommender) moves first, the MAX player (adversary) second. The leaves of the minimax tree are labeled with the pairwise max regret of the two choices on its path.

A full evaluation of the tree requires the solution of $n(n-1)$ pairwise regret LPs (noting that the MIN player's choice need not be explicitly evaluated or even represented as a possible MAX choice, since it must yield pairwise regret of 0). However, it is generally not necessary to evaluate every node of the tree as *Alpha-beta pruning* (see (Russell and Norvig 2003) for an introductory description) can be used to eliminate branches from evaluation.

[3]The choice of the utility function by the adversary is dictated by pair of options, so it need not be modeled as a move.

Alpha-beta pruning is simple in such a simple game tree: during the tree evaluation, we maintain an upper bound UB (initially $+$Inf) at the root, representing the max regret of the best solution found so far (from the perspective of MIN), and lower bounds $LB(n)$ at each MAX node, one for each possible MIN choice (or recommendation). Every time we evaluate a leaf node, we compute pairwise regret $MR(o_{\min}, o_{\max}, W)$ of MIN's choice $o_{\min}$ and MAX's choice $o_{\max}$ on the path. We update the lower bound at the corresponding MAX node, and prune (α cut [4]) whenever $LB(n) \geq UB$. This is because $MR(o_{\min}, o_{\max}, W) \leq MR(o_{\min}, W)$. At the same time, whenever we complete the evaluation of a MAX node n, we update the upper bound UB to $\min(UB, v(n))$ where $v(n)$, the value of the node, is the maximum value among the leafs.

The efficiency of this pruning depends on the order in which nodes are evaluated (Russell and Norvig 2003); this is especially true given the very shallow, broad nature of our tree. Pruning is most effective when, at each node, the best children (with respect to the relevant node evaluation, MIN or MAX) are evaluated first. Figure 2 shows, in our simple example, that in the best case only 5 nodes out of 9 need to be evaluated (i.e., 5 pairwise regret maximization). To speed up the search, we consider a heuristic that first evaluates choices at the MIN (recommender) node that are likely to be good candidates for minimizing max regret; and we first evaluate at at MAX (adversarial) nodes options that are likely to induce high regret against the given MIN choice. These heuristics give us an evaluation order for both MIN and MAX choices and can lead to considerable pruning. We discuss each in turn.

For the MIN node, we note that the regret of any option is maximized at one of the vertices of the feasible region W. Thus we sample t vertices (for instance, by considering extreme weights that maximize the importance of one of the attributes) and refer to the the $\mathbf{w}$ so-sampled as *reference utilities*. These are used to initialize the lower bounds $LB(n)$: we simply compute the actual regret with respect to these utilities for the option that leads to (MAX node) n. We then evaluate MIN's children n in increasing order of initial lower bound $LB(n)$.

To order the children of MAX node, for each MAX node n, we consider the feasible utility function $\mathbf{w}^-$ that minimizes the utility the MIN choice. (This requires a simple optimization.) The option that maximizes utility at $\mathbf{w}^-$ (i.e., the optimal choice under $\mathbf{w}^-$) is likely to give a high value of for pairwise regret and thus represents a potentially good adversary. Moreover, once we have generated $\mathbf{w}^-$, we can use it to update the lower bound by considering the actual regret for each option. MAX choices are evaluated in order of decreasing utility under $\mathbf{w}^-$.

In practice, these heuristics can significantly speed up the computation of minimax regret in product databases. Table 1 shows that number of pairwise regret checks (LPs) is almost linear in the number of options in the database; indeed, with these orderings, MAX nodes are often pruned immediately without even considering an adversarial choice. (These are experiments run on synthetic data for illustrative purposes.)

[4] beta cuts are not possible given the depth of the tree

size	attributes	constraints	num of pairwise checks
40	4	0	41
200	5	0	207
400	7	10	492
1000	10	0	1003
1000	10	60	1998
1000	15	30	999

Table 1: Number of pairwise regret checks to compute minmax regret on some sample datasets. We evaluate the search tree with our heuristics of reference utilities.

Set Recommendations: Setwise Minimax Regret

We now consider the modification of the techniques above for setwise minimax regret. Naturally, setwise max regret is more computationally demanding, requiring selection of a set of options. However, it is still possible to formulate the computation in a MIP for configuration problems. Database problems are more challenging: the adversarial search presented above for single-item recommendation can be applied directly, with replacement of a single move by the recommender (MIN player) by k moves, corresponding to the choice of k options for the slate. However, performance can take a dramatic hit as the size of the desired recommendation set increases. However, we develop a simple heuristic hill-climbing strategy that seems to provide very good recommendation sets in practice.

Configuration problems: MIP formulation For configuration problems we formulate the problem of setwise minimax regret following the general strategy for single-option minimax regret, formulating as a (MIP) minimization with exponentially many constraints. We use a constraint generation procedure to prevent enumeration of the entire constraint set (Boutilier *et al.* 2006a; 2004). However, there are some critical differences in the formulation, which we describe here.

Setwise minimax regret for configuration problems can be formulated as the following MIP.

$$\min_{M, I_w^j, \mathbf{X}^j, V_w^j} \quad M$$

$$\text{s.t. } M \geq \sum_{1 \leq j \leq k} V_{\mathbf{w}}^j \quad \forall \mathbf{w} \in \mathit{Vert} \tag{10}$$

$$V_w^j \geq \mathbf{w} \cdot (\mathbf{x}_{\mathbf{w}}^* - \mathbf{X}^j) + (I_w^j - 1) m_{big} \tag{11}$$

$$\forall j \in [1,k] \wedge \quad \forall \mathbf{w} \in \mathit{Vert} \tag{12}$$

$$\sum_{1 \leq j \leq k} I_{\mathbf{w}}^j = 1 \quad \forall \mathbf{w} \in \mathit{Vert} \tag{13}$$

$$I_{\mathbf{w}}^j \in \{0,1\} \tag{14}$$

$$V_{\mathbf{w}}^j \geq 0 \quad \forall j \in [1,k], \forall \mathbf{w} \in \mathit{Vert} \tag{15}$$

This MIP minimizes M by: (a) choosing k options (or configurations $\mathbf{x}_j$ designated by variables $\mathbf{X}^j$ (where each $\mathbf{X}^j$ is a vector of n attributes) for the recommendation set;

(b) selecting, for each adversary w, one of those options (the jth option) as the choice that has minimum max regret against an adversary, and ensuring that M is greater than the true regret of the jth option relative to every possible choice of adversary utility function and option.

Note however that this constraint need not be applied to (continuously many) utility functions or exponentially many adversarial choices. In the MIP, we post these constraints only for each vertex of W (i.e., $\mathbf{w} \in Vert(W)$) and for the optimal product choice $\mathbf{x}^*_{\mathbf{w}}$ for that vertex. This relies on the observation that regret maximized at vertices of W, and, for any adversarial choice of $\mathbf{w}$, the adversarial option that maximizes the pairwise regret for any user choice is the optimal option for $\mathbf{w}$.[5]

However, this MIP still requires (potentially) exponentially many constraints, one for each element of $Vert(W)$. We can make computation much more effective by applying constraint generation, observing that at the optimal solution, very few of these constraints are likely to be active. Our procedure works as follows: we solve a relaxed version of the MIP above—the *master problem*—using only the constraints corresponding to a small subset $Gen \subset Vert(W)$ of the constraints in the MIP above. We then test whether any unexpressed constraints are violated at the current solution. This involves computing the true setwise max regret of the slate generated by the master problem. If the true setwise max regret is of the slate is greater than δ, we know that a constraint has been violated. Specifically, the computation of setwise max regret will produce the element $\mathbf{w} \in Vert(W)$ and optimal product $\mathbf{x}^*_{\mathbf{w}}$ that corresponds to the maximally violated constraint at the current master solution. So if a constraint is violated, we add this maximally violated constraint to Gen, tightening the MIP relaxation, and repeat; if not, we are assured that the current solution minimizes setwise max regret.[6]

In the formulation, m_{big} is an arbitrary big number, that we need to encode the fact that, for any given w, only the option with the highest utility (among those in the slate) with respect to w contributes to the actual setwise regret.

The SMR maximization subproblem can be also encoded with a MIP, similar to (Boutilier *et al.* 2006b). The optimization makes use of a decision variable to explicitly represent the setwise regret, M, to be maximized and we constrain M to be greater than the single max regret $MR(\mathbf{x}_j, W)$, for each option in the slate.

Database Problems: A Hill-climbing Strategy As discussed above, while minimax search can be applied directly to the problem of setwise minimax regret for database problems, scaling is sometimes a concern. We now present a heuristic hill-climbing strategy that scales much more effectively. We describe it in the context of database problems, but it can also be used directly for configuration problems.

The central idea is that is possible to modify a given recommendation set $\mathbf{Z}$ in such a way that setwise max regret cannot increase, and usually decreases until a high quality set is found. We define the *MMR-transformation* T to be a mapping that refines a recommendation set $\mathbf{Z}$ by partitioning the current feasible utility space W into $\{W^{\mathbf{Z}\rightarrow\mathbf{x}_i}\}, \forall \mathbf{x}_i \in \mathbf{Z}$, as discussed in Observation 1. In each partition we compute the *single* recommendation that has minimax regret in that region of utility space, and define the new set recommendation $T(\mathbf{Z})$ to be the collection of these (single) minimax-optimal recommendations.

Definition 2 *Define the* MMR-transformation $T : \mathbf{Z} \rightarrow \mathbf{Z}'$, *where* $\mathbf{Z} = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$, *to be* $T(\mathbf{Z}) = \{\mathbf{x}'_1, \ldots, \mathbf{x}'_k\}$ *such that for all* $1 \leq i \leq k$*:*

$$\mathbf{x}'_i = MMR\text{-}Opt(W^{\mathbf{Z}\rightarrow\mathbf{x}_i})$$

We can show that T cannot increase setwise max regret.

Observation 2 *For any set recommendation* $\mathbf{Z}$ $SMR(T(\mathbf{Z})) \leq SMR(\mathbf{Z})$

We use the MMR-transformation to define our heuristic search strategy to produce good recommendation sets; intuitively, we repeatedly T until a fixed point (with respect to setwise max regret, not the set itself) is found.

Alg 1 Hill-climbing-T algorithm (HCT)
The algorithm considers an initial set $\mathbf{Z}$*, and rewrites* $\mathbf{Z}$ *using* T *until a fixed-point is found.*

- **Repeat** *Z := T(Z)*
- **Until** $SMR(T(\mathbf{Z}), W) = SMR(\mathbf{Z}, W)$

We initialize the slate $\mathbf{Z}$ using the current solution strategy (CSS), empirically, this seems to produce the most promising recommendation sets. For $k = 2$, this means that the initial set is $\mathbf{Z} = \{x^*_W, x^w\}$, where $x^* = MMR\text{-}Opt(W)$, and $x^w = MRAdv(W)$.

For larger sets ($k > 2$), there is not a standard definition of the CSS. We propose to use the following strategy, that we call *chain of adversaries*, to generate the initial slate. We start from $\{x^*_W, x^w\}$ and repeatedly maximize setwise max regret given the current set, in some sense maximizing the diversity of choices from perspective of utility space. This gives the set $\{\mathbf{x}^1, \ldots, \mathbf{x}^k\}$ where:

$$\begin{cases} \mathbf{x^1} = & \mathbf{x}^*_W \\ \mathbf{x^i} = & Adv(\{\mathbf{x^1}, .., \mathbf{x^{i-1}}\}, W) \;\; 2 \leq i \leq k \end{cases}$$

The chain of adversaries requires to solve single minimax regret once, and then $k - 2$ setwise regret maximizations. The chain of adversaries can be seen as a generalization of CSS to sets of any size, and could also be considered as an alternative, faster strategy to select recommendations.

[5] $V^j_{\mathbf{w}}$ is the actual regret of option $\mathbf{X}^j$ of the slate with respect to the utility $\mathbf{w}$ when the corresponding $I^j_{\mathbf{w}}$ is activated. For any $\mathbf{w}$, one and only one $I^j_{\mathbf{w}}$ is set to 1. In order to minimize M, the optimization will activate the $I^j_{\mathbf{w}}$ corresponding to the $\mathbf{x}^j$ with lower actual regret. The first constraint (10) captures the idea that for a slate of options, given $\mathbf{w}$, the regret of the joint slate is the minimum among the individual values of regret (in the summation, all but one term are zeros).

[6] Note that the adding a new constraint requires the introduction of new variables to the master problem. Every time we add a new w to Gen, k new variables I and V are necessary.

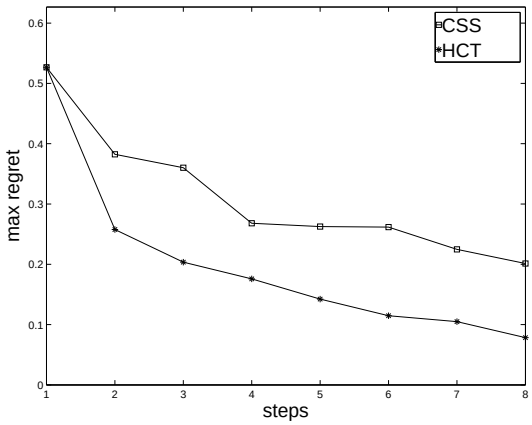


Figure 3: The hillclimbing strategy based on setwise regret outperforms the current solution strategy in this experiment (20 runs).

Myopic Elicitation

In addition to produce final recommendations, our criterion can also be used as a driver for further elicitation of the utility function of the user. In fact, whenever we consider a slate of recommendations, the user may give us some feedback, perhaps selecting the option that she prefers among those in the set. This information is very valuable, and as we have seen in the initial example, can be used to reduce regret. It is therefore interesting to assess the value of a recommendation set also with respect to the possible feedback.

An important observation is that in the case of comparison queries (the user selects the preferred option in a slate), the set of k optimal recommendations that minimize setwise regret is also the optimal choice set for a comparison query with respect to *myopic worst case regret* (WR), a measure of the value of information of a query.

The *Worst-case Regret* (WR) of a comparison query based on a choice set $\mathbf{Z} = \{\mathbf{x}_1, .., \mathbf{x}_k\}$ is defined as

$$WR(\{\mathbf{x}_1, .., \mathbf{x}_k\}) = max[MMR(W^{Z \to \mathbf{x}_1}), .., MMR(W^{Z \to \mathbf{x}_k})] \quad (16)$$

WR considers the "single" max regret in each possible scenario. It is possible to verify that $WR(\mathbf{Z}) \leq SMR(\mathbf{Z})$; the worst case regret is always lower (or equal) than the setwise max regret.

The optimality of minimax setwise recommendations (we omit the full proof for reasons of space) with respect to WR is based on the consideration (an extension of Observation 2) that the transformation T introduced in the previous section is also such that $SMR(T(\mathbf{Z})) \leq WR(\mathbf{Z})$ (the proof requires considering the different partitions imposed by Observations 1 and compare the two expression componentwise). We call $\mathbf{Z}^*$ the optimal recommendation set according to setwise regret. A set $\mathbf{Z}'$ such that $WR(\mathbf{Z}') < WR(\mathbf{Z}^*)$ but $SMR(\mathbf{Z}') > SMR(\mathbf{Z}^*)$ leads to contradiction.[7]

We performed some preliminary experiments in order to evaluate our recommendation strategy from the prospective of elicitation. We are interested in quantifying the reduction of regret in practice. In Figure 3 we compare the efficiency of an elicitation based on our SMR criterion and the current solution strategy (CSS), considering a syhnthetic dataset with 5000 options and 10 attributes. We plot max regret in function of the number of queries (steps). SMR is optimized using the hill-climbing strategy (HCT).

Example Critiquing

As an evaluation setting, we apply our regret-based recommender to example-critiquing. This domain is interesting because current systems usually rely on heuristics and we expect that an utility-based approach can be greatly beneficial.

Critiquing is a setting where the user expresses feedback on options that the system shows to her. In particular we consider a particular version of critiquing, often called the *dynamic critiquing* model (Reilly *et al.* 2005), where a *current* product or recommendation is displayed, and the user is invited to move to a different product by choosing particular actions (laid out in the interface) that change the product. They can include *unit critiques*, which request modification of a particular product attribute; e.g., "give me a laptop that is *lighter* than the current one."

Often alternative *suggestions* or *compound critiques* are used in which multiple attributes ("lighter and faster processor, but more expensive") are tweaked, or in which a selection is made from a system-suggested set of alternative products ("let me see laptop 3 instead of the current one").

The set of possible critiques is generated by the system, and the user chooses one of the possible actions. At each interaction, the user may choose to critique the current product if she is not completely satisfied with it, or simply because she wishes to explore the product space in more depth.

In general those systems use heuristics to generate the set of possible critiques. However, we expect that better performance can be obtained if critiquing suggestions are selected according to a decision-theoretically sound criterion as our setwise minimax regret.

In order to implement our approach, it is necessary to give a precise semantics to each of the critiquing actions. We identify two main reasons a user will critique an option. First, she may want to explore the product space in an effort to better understand either the space of feasible options or her own preferences. This latter desire makes sense especially when one adopts the view commonly held in behavioral economics that decision support systems should help people *construct* their preferences (not just articulate them) (Slovic 1995). Second, she may wish to improve the current product, making tradeoffs among her preferences for different attributes. It is this latter *exploitive* or *improvement mode* that critiquing systems fail to account for adequately when deciding on appropriate product suggestions. In this evalua-

[7] If we apply the transformation T to $\mathbf{Z}'$ we obtain a set $\bar{\mathbf{Z}}$ such that $SMR(\bar{\mathbf{Z}}) \leq WR(\mathbf{Z}') < WR(\mathbf{Z}^*) \leq SMR(\mathbf{Z}^*)$ but this means that $SMR(\bar{\mathbf{Z}}) < SMR(\mathbf{Z}^*)$, contradicting the optimality of $\mathbf{Z}^*$ with respect to SMR.

tion we use critiques of the latter type to constrain the set of possible user utility functions.

In the following, we describe our simulation setting and present our results.

Experiments

To validate our regret-based approach to critiquing we designed a framework that simulates a full interaction of a user with a user interface. As in a real system, each simulation comprises a number of cycles of interaction, each showing a current product which the user can critique using either unit or the selection of one of the suggested recommendations.

The simulated user continues the critiquing process until the perceived increase in utility is lower than some threshold. We assume that among all possible critiquing actions, the one with highest perceived improvement will be chosen by the user.

In our experiment, at each interaction the system displays:

- the *current product*,
- a choice of *unit critiques* of the current product (they request the modification of a particular product attribute),
- a set of *suggestions*, alternative options that can change focus for the search, presented as such or labeled as *compound* critiques ("lighter and faster processor, but more expensive")

At each step, the user can choose to either select the current product (and finish the interaction) or to tweak it in order to improve it and get better recommendations in the next cycle.

We compare our regret-based approach to three other approaches that use compound critiques. In our case, we use the generation of a set of recommendations (based on setwise regret) to display as alternatives. One is selected as *current product* and the others are displayed as *suggestions*.

We briefly review the different critiquing approaches and then we present the experimental results.

Dynamic Critiquing

The *dynamic critiquing* model (Reilly *et al.* 2004) makes use of a particular similarity metric to retrieve the current product and uses the APriori datamining algorithm to propose alternative compound critiques. The algorithm dynamically generates compound critiques by discovering common feature patterns among the set of products. Essentially, each compound critique describes a set of products in terms of the features they have in common. For example in the PC domain, a typical compound critique might be "Faster CPU and a Larger Hard Drive." Whenever a product is shown to the user as the current product, the APriori datamining algorithm is used to quickly discover these patterns and convert them into a set of suggested compound critiques. Each compound critique corresponds to a product that is, among all products satisfying the pattern most similar to the current one.

The generation of suggestions consists of two steps. First, each product is matched against the current product to produce lists of *critique patterns*, each comprising an attributes and a comparison operators from the set: $<$,$>$,$\neg$,$=$. An example pattern might be: {[Price $>$], [ProcessorSpeed $>$]}. Second, the algorithm uses APriori to find recurrent critiquing patterns; a compound critique based on a pattern is then presented to the user it has sufficient support in the product database. In our experiments, the support threshold is set to 0.3 and selection of compound critiques corresponds to the *low-support* strategy in (Reilly *et al.* 2004).

Incremental Critiquing

Incremental critiquing (Reilly *et al.* 2005) (IC) improves the basic dynamic critiquing model by incorporating a user model. While suggestions are still based on the APriori algorithm (as above), the retrieval of the next product associated with a critiquing action is based on a *quality metric* that values both the $score$ given to the product by the preference model and its similarity to the current product.

In the implementation we developed for our experiments, we take advantage of the fact that the preference ordering over attributes is known: the score is dictated by a linear utility function that gives equal weight to all attributes. The initial product is the option with maximum utility. When retrieving the next example from the set of products that satisfy the user-chosen critique, we select the product $\mathbf{x}$ that maximizes $score(\mathbf{x}) \cdot Similarity(\mathbf{x}, \mathbf{y})$, where $\mathbf{y}$ is the product recommended at the previous cycle, and $score$ is the heuristic utility function.

Incremental Critiquing: MAUT

Another implementation of incremental critiquing (Reilly *et al.* 2007) uses a simple multi attribute utility (MAUT) model to make recommendations and generate compound critiques (rather than similarity). In this approach, a simple additive utility model u is generated, initially giving equal weight to all attributes; each time an attribute is critiqued, its weight is multiplied by a constant (and all weights renormalized). The original design of this algorithm makes use of parameterized value functions for each attribute, where the value taken by the current option is considered preferred. Since our experimental set up assumes that the local preference ordering over attribute values is known, we instead assume a linear utility model.

Suggestions are generated using optimization with respect to the estimated utility model, and the k best products are presented as alternative cases. A limitation of this approach is its reliance on a fixed utility model (as opposed to reasoning with the space of possible user utilities). Moreover, options that *all* have high value in a single utility sample are unlikely to be diverse or informative enough to generate useful distinctions.

Regret-based critiquing

Our version of dynamic critiquing exploits setwise minimax regret using the ideas above. Specifically, at any point in the interaction cycle, we generate the current optimal recommendation set (with respect to minimax setwise regret), and propose one of these options a current product. The remainder of the set is used to display suggestions.

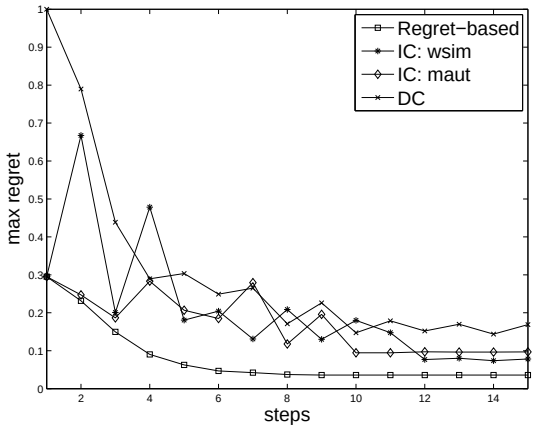


Figure 4: Maximum regret of the recommended option at each step for four algorithms: regret-based critiquing (regret), dynamic critiquing with compound critiques generated with APriori (DC), incremental critiquing with weighted similarity and APriori (IC) and incremental critiquing with a multiattribute utility model (IC maut).

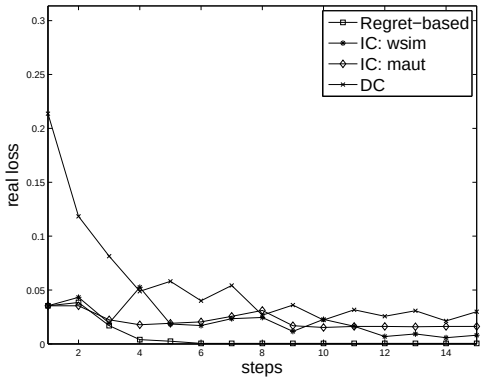


Figure 5: True regret (real loss) of the recommended option at each step for four algorithms: regret-based critiquing (regret), dynamic critiquing with compound critiques generated with APriori (DC), incremental critiquing with weighted similarity and APriori (IC) and incremental critiquing with a multiattribute utility model (IC maut).

Empirical Results

In the experiments we compare the four different versions of dynamic critiquing discussed above: the original dynamic critiquing algorithm (similarity plus APriori), incremental critiquing, incremental critiquing with MAUT, and regret-based critiquing. We used a C implementation of the APriori algorithm (Bodon 2003). All systems make available unit critiques of any attribute (and user's adopt an expected improvement semantics). We evaluate the performance of all algorithms with respect to recommendation efficiency and offer some speculative examination of regret-based critiquing in terms the tradeoff between cognitive cost and number of compound critique options presented at each interaction.

We evaluate the different critiquing methods by comparing the quality of the recommendations with respect to max regret. We tested the methods on a real database of 200 apartments, using randomly drawn utility functions (as described above), and $k = 3$ suggested products at each interaction cycle. All results are averaged over 20 simulated users. Fig. 4 shows the maximum regret of the recommended product at each stage of the interaction. We note that regret-based critiquing outperforms the other methods of generating compound critiques by a wide margin. This is true when considering both the "anytime" profile of the method (i.e., the degree to which minimax drops) and its final convergence: our technique converges on a product who max regret is about 3% on average, while the MAUT incremental critiquing settles at about 10% (and the others worse, with dynamic critiquing unable to reduce max regret to less than 18%).

More interesting is the fact that regret-based critiquing offers better "actual" recommendations, as measured by true regret (difference from the true optimal recommendation). Regret-based critiquing is designed to attack *bounds* on regret (i.e., worst-case loss); so one might wonder whether other techniques find better products despite being unable to "prove" that they are good. Fig. 5 shows this not to be the case. While other critiquing techniques recommend products that are much better than their regret-bounds suggest, regret-based critiquing is able to consistently find the optimal product (and find a near-optimal product in as few as five or six interaction cycles). By contrast, the other three methods are unable to identify the optimal option at convergence.

Conclusions

In this paper we presented a novel formalization of recommendations of a joint set of alternatives based on the notion of regret. The criterion that we propose, setwise max regret, represents an intuitive extension of the traditional regret criterion for single recommendations.

We show how optimal recommendation sets (with respect to our criterion) can be computed with mixed integer programming (MIP) methods and the constraint generation technique when options are constructed from a set of configuration constraints. Alternatively, set recommendations can be obtained using a hill-climbing strategy interleaved with adversarial search in discrete settings.

We discuss the problem of utility elicitation, showing that our recommendation strategy reduces max regret more quickly than any other possible choice. Finally we present an application of these principles for critiquing systems.

Our reliance on explicit utility modeling and minimax regret provides a powerful new means of generating good critiques and making good product recommendations. Our regret-based critiquing recommender can often lead to optimal recommendations using very few, say, compound critiquing interactions, and outperforms other dynamic critiquing techniques both in speed of convergence and the quality of the final recommendations.

The incorporation of noisy feedback is an important next

step; we are currently considering the possibility of a clarification dialogue. The idea is to verify information that is sensitive with respect to regret.

Largely unaddressed in our critiquing model is the need for users to explore the product space, one of the main advantages of critiquing. We are currently developing hybrid models in which the system and/or user explicitly distinguishes exploratory actions from improving actions. Even with such a distinction, there is still the interesting question of modeling user *search processes* in a way that would allow insight into preferences to be drawn during exploration as well. Finally, the development of models of cognitive costs using techniques from behavioral economics, decision theory and psychology remains an important avenue of future research.

References

Fahiem Bacchus and Adam Grove. Graphical models for preference and utility. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence (UAI-95)*, pages 3–10, Montreal, 1995.

Ferenc Bodon. A fast apriori implementation. In Bart Goethals and Mohammed J. Zaki, editors, *Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'03)*, volume 90 of *CEUR Workshop Proceedings*, Melbourne, Florida, USA, 19. November 2003.

Craig Boutilier, Fahiem Bacchus, and Ronen I. Brafman. UCP-Networks: A directed graphical representation of conditional utilities. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI-01)*, pages 56–64, Seattle, 2001.

Craig Boutilier, Richard S. Zemel, and Benjamin Marlin. Active collaborative filtering. In Christopher Meek and Uffe Kjærulff, editors, *UAI*, pages 98–106. Morgan Kaufmann, 2003.

Craig Boutilier, Tuomas Sandholm, and Rob Shields. Eliciting bid taker non-price preferences in (combinatorial) auctions. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-04)*, pages 204–211, San Jose, CA, 2004.

Craig Boutilier, Relu Patrascu, Pascal Poupart, and Dale Schuurmans. Constraint-based optimization and utility elicitation using the minimax decision criterion. *Artifical Intelligence*, 170(8–9):686–713, 2006.

Craig Boutilier, Relu Patrascu, Pascal Poupart, and Dale Schuurmans. Constraint-based optimization and utility elicitation using the minimax decision criterion. *Artif. Intell.*, 170(8-9):686–713, 2006.

Craig Boutilier. A POMDP formulation of preference elicitation problems. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-02)*, pages 239–246, Edmonton, 2002.

Darius Braziunas and Craig Boutilier. Preference elicitation and generalized additive utility. In *AAAI*, 2006.

Darius Braziunas and Craig Boutilier. Minimax regret-based elicitation of generalized additive utilities. In *Proceedings of the Twenty-third Conference on Uncertainty in Artificial Intelligence (UAI-07)*, pages 25–32, Vancouver, 2007.

Darius Braziunas and Craig Boutilier. Minimax regret based elicitation of generalized additive utilities. In *Proceedings of the Twenty-third Conference on Uncertainty in Artificial Intelligence (UAI-07)*, pages 25–32, Vancouver, 2007.

Urszula Chajewska and Daphne Koller. Utilities as random variables: Density estimation and structure discovery. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI-00)*, pages 63–71, Stanford, 2000.

Urszula Chajewska, Daphne Koller, and Ronald Parr. Making rational decisions using adaptive utility elicitation. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-00)*, pages 363–369, Austin, TX, 2000.

Peter C. Fishburn. Interdependence and additivity in multivariate, unidimensional expected utility theory. *International Economic Review*, 8:335–342, 1967.

Ralph L. Keeney and Howard Raiffa. *Decisions with Multiple Objectives: Preferences and Value Trade-offs*. Wiley, New York, 1976.

Panos Kouvelis and Gang Yu. *Robust Discrete Optimization and Its Applications*. Kluwer, Dordrecht, 1997.

David McSherry. Diversity-conscious retrieval. In *ECCBR '02: Proceedings of the 6th European Conference on Advances in Case-Based Reasoning*, pages 219–233, London, UK, 2002. Springer-Verlag.

Robert Price and Paul R. Messinger. Optimal recommendation sets: Covering uncertainty over user preferences. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI'05)*, pages 541–548, 2005.

James Reilly, Kevin McCarthy, Lorraine McGinty, and Barry Smyth. Dynamic critiquing. In Peter Funk and Pedro A. González-Calero, editors, *ECCBR*, volume 3155 of *Lecture Notes in Computer Science*, pages 763–777. Springer, 2004.

James Reilly, Kevin McCarthy, Lorraine McGinty, and Barry Smyth. Incremental critiquing. *Knowl.-Based Syst.*, 18(4-5):143–151, 2005.

James Reilly, Jiyong Zhang, Lorraine McGinty, Pearl Pu, and Barry Smyth. Evaluating compound critiquing recommenders: a real-user study. In Jeffrey K. MacKie-Mason, David C. Parkes, and Paul Resnick, editors, *ACM Conference on Electronic Commerce*, pages 114–123. ACM, 2007.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition edition, 2003.

Ahti Salo and Raimo P. Hämäläinen. Preference ratios in multiattribute evaluation (PRIME)–elicitation and decision procedures under incomplete information. *IEEE Trans. on Systems, Man and Cybernetics*, 31(6):533–545, 2001.

Leonard J. Savage. *The Foundations of Statistics*. Wiley, New York, 1954.

Paul Slovic. The construction of preference. *American Psychologist*, 50(5):364–371, 1995.

Barry Smyth and Paul McClave. Similarity vs. diversity. In David W. Aha and Ian Watson, editors, *ICCBR*, volume 2080 of *Lecture Notes in Computer Science*, pages 347–361. Springer, 2001.

Uncovering Functional Dependencies in MDD-Compiled Product Catalogues

Tarik Hadžić and **Barry O'Sullivan**
Cork Constraint Computation Centre
Department of Computer Science, University College Cork, Ireland
{t.hadzic|b.osullivan}@4c.ucc.ie

Abstract

Product catalogues are usually represented as tables. They are regularly updated with new variants and, over time, various forms of inconsistencies or undesirable properties can be introduced, especially when global changes are made. We argue that by compiling product catalogues into decision diagrams we can support a number of high-level queries for detecting and checking for various forms of inconsistency, as well as verifying other properties relevant for user interaction. In particular, a class of functional dependency-based inconsistencies can be detected efficiently. An example is verifying properties such as: *"each identical configuration of a product has the same price"*. This paper presents a number of algorithmic advances. We illustrate the usefulness of our approach by evaluating these high-level properties over real-world publicly available product catalogues.

1 Introduction

Uncovering functional dependencies is an important problem in many artificial intelligence (AI) domains. Many AI datasets are represented in tabular form, defined in terms of a set of attributes (columns). A large dataset might be represented in a database table or spreadsheet and determining that one or more attributes is functionally determined by values of other attributes can be of critical importance, e.g. in analyzing the effect of chemical compounds on cancerogenity or studying the shopping habits of the customers. Functional dependencies are most commonly used in the process of designing relational database schema.

We suggest that uncovering functional dependencies can also be useful in an online product configuration system context. For example, various forms of *catalogue consistencies* can be expressed as functional dependencies. For example, checking whether "each identical configuration of a product has the same price" reduces to verifying that the price attribute is functionally determined by the remaining attributes. This could be relevant for product catalogues that are regularly updated with new variants, or preprocessed for various purposes. Furthermore, functional dependencies can help to reason about the length of user interaction with respect to the design of the user interface. If a small subset of attributes functionally determines all the others, then an interface that encourages a user to first assign attributes from such a subset, might help reduce the total number of interaction steps.

In this paper we study the problem of identifying and testing functional dependencies amongst subsets of the attributes X that define a catalogue. Specifically, we consider dependencies of the form $Y \rightarrow x$ for a subset of variables $Y \subset X$ and variable $x \in X$, which hold if and only if for any two products p_1, p_2 from the catalogue, whenever p_1 and p_2 agree on attributes Y, i.e. $p_1[Y] \equiv p_2[Y]$, they also agree on attribute x, i.e. $p_1[x] \equiv p_2[x]$.

A number of approaches have been suggested in the database community for uncovering functional dependencies [Huhtala *et al.*, 1999]. However, all of them find dependencies by operating over an explicit representation of the catalogue, and state-of-the art approaches take linear time, $\mathcal{O}(T)$, in the number of products, T, in the catalogue [Huhtala *et al.*, 1999; Schlimmer, 1993]. Other approaches incur even more complexity by using sorting, in $\mathcal{O}(T \cdot log(T))$ time, or explicit comparison of all tuples, in $\mathcal{O}(T^2)$ time. Note that the number of tuples can grow exponentially with the number of attributes, however most real world catalogues are very sparse and contain only a tiny proportion of all possible products.

The main contribution of this paper is an approach to uncovering functional dependencies when a dataset is compressed into a *multi-valued decision diagram* (MDD) rather than as an extensionally represented table, the standard in relational database theory. MDDs are directed acyclic graphs that, for a sorted list of attributes, use prefix and suffix sharing to compactly represent the data. Each product is encoded as a path, and every edge on the path encodes an attribute-value pair (variable assignment), $x_i = a$. While the size of an MDD is, in the worst-case, linear in the number of products, they can often be *exponentially* smaller. We propose a number of algorithms for uncovering functional dependencies; these algorithms have time complexity that is quadratic in the size of the MDD. In worst case, when there is no compression, we get $\mathcal{O}(T^2)$ running time. However, whenever the MDD is small we guarantee a sublinear-time algorithm for testing $Y \rightarrow x$.

We implemented the algorithms we have introduced in this paper and applied them to several publicly available product

catalogues. As a result, we have uncovered several properties, some of which indicate "bugs" in the datasets which have previously gone undetected.

The remainder of this paper is organised as follows. Section 2 presents the necessary technical background required throughout the paper. We show how functional dependencies can be uncovered in decision diagram representations of product catalogues in Section 3. A variety of specialised functional dependencies, called subset-induced dependencies are presented in Section 4. We define the notion of approximate dependencies in Section 5, which are inexact forms of standard functional dependencies. We report on a number of experiments in Section 6. Section 7 presents a number of concluding remarks and outlines directions for future work.

2 Background

In this section we provide the necessary background on preliminary concepts and introduce our notational conventions.

2.1 Solution Sets

We are given a set of variables (product attributes) $X = \{x_1, \ldots, x_n\}$ defined over finite domains $D_1, \ldots, D_n$ of possible values of the attributes of a product. We are also given a set of solutions (available products) $Sol \subseteq D_1 \times \ldots, D_n$, which can be defined either explicitly, e.g. as a set of items in a product catalogue, or implicitly, e.g. as the set of solutions to a constraint satisfaction problem $\langle X, D, F =_{\text{def}} \{c_1, \ldots, c_m\}\rangle$ defining which products can be manufactured in a factory. The latter approach is common when defining a catalogue of configurable products. Consider for example the following implicit representation of a T-shirt catalogue.

Example 1 (Representing a T-Shirt Catalogue)**.** *We are interested in selecting a T-shirt which is defined by three attributes: the color (black, white, red, or blue), the size (small, medium, or large) and the print ("Men In Black" - MIB or "Save The Whales" - STW). There are two rules that define the set of valid combinations: if we choose the MIB print then the color black has to be chosen as well, and if we choose the small size then the STW print (including a big picture of a whale) cannot be selected as the picture of a whale does not fit on the small shirt. The implicit representation (X, D, F) of the T-shirt example consists of variables $X = \{x_1, x_2, x_3\}$ representing color, size and print. Variable domains are $D_1 = \{0, 1, 2, 3\}$ $(black, white, red, blue)$, $D_2 = \{0, 1, 2\}$ $(small, medium, large)$, and $D_3 = \{0, 1\}$ (MIB, STW). The two rules translate to $F = \{f_1, f_2\}$, where f_1 is $x_3 = 0 \Rightarrow x_1 = 0$ $(MIB \Rightarrow black)$ and f_2 is $(x_2 = 0 \Rightarrow x_3 \neq 1)$ $(small \Rightarrow not\ STW)$.* ♢

The same solution space of the T-shirt example (satisfying the configuration rules F) can be also given explicitly as a set of items in a catalogue, as shown in Table 1.

Table 1: Solution set for the T-shirt example.

color	size	print
black	*small*	*MIB*
black	*medium*	*MIB*
black	*medium*	*STW*
black	*large*	*MIB*
black	*large*	*STW*
white	*medium*	*STW*
white	*large*	*STW*
red	*medium*	*STW*
red	*large*	*STW*
blue	*medium*	*STW*
blue	*large*	*STW*

2.2 Decision Diagrams

Decision diagrams are compressed representations of solution sets $Sol \subseteq D_1 \times \ldots \times D_n$. Formally, decision diagrams are a family of rooted directed acyclic graphs (DAGs) where each node u is labeled with a variable x_i and each of its outgoing edges e are labeled with a value $a \in D_i$. The decision diagram contains one or more *terminal* nodes, each labeled with a constant and having no outgoing edges. The most well known member of this family is the *binary decision diagram* (BDD) [Bryant, 1986] which is used as a compressed representation of Boolean functions in many areas, such as verification, model checking, VLSI design [Meinel and Theobald, 1998; Wegener, 2000; Drechsler, 2001], etc. In this paper we will primarily operate with the following variant, called *multi-valued decision diagrams*:

Definition 1 (Multi-valued Decision Diagram)**.** *An MDD M is a rooted directed acyclic graph (V, E), where V is a set of vertices containing the special terminal vertex $\mathbf{1}$ and a root $r \in V$. Further, var : $V \to \{1, \ldots, n+1\}$ is a labeling of all nodes with a variable index such that var$(\mathbf{1}) = n + 1$. Each edge $e \in E$ is denoted with a triple (u, u', a) of its start node u, its end node u' and an associated value a.*

We work only with *ordered* MDDs. A total ordering $<$ of the variables is assumed such that for all edges (u, u', a) var$(u) <$ var(u'). For convenience we assume that the variables in X are ordered according to their indices. Ordered MDDs can be considered as being arranged in n *layers* of vertices, each layer being labeled with the same variable index. We will denote as V_i the set of all nodes labeled with x_i, $V_i = \{u \in V \mid \text{var}(u) = i\}$. Similarly, we will denote with E_i the set of all edges originating in V_i, i.e. $E_i = \{e(u, u', a) \in E \mid \text{var}(u) = i\}$. Unless otherwise specified, we assume that on each path from the root to the terminal, every variable labels exactly one node. An MDD encodes a CSP solution set $Sol \subseteq D_1 \times \ldots \times D_n$, defined over variables $\{x_1, \ldots, x_n\}$. To check whether an assignment $\mathbf{a} = (a_1, \ldots, a_n) \in D_1 \times \ldots \times D_n$ is in Sol we traverse M from the root, and at every node u labeled with variable x_i, we follow an edge labeled with a_i. If there is no such edge then $\mathbf{a}$ is not a solution $\mathbf{a} \notin Sol$. Otherwise, if such a traversal eventually ends in terminal $\mathbf{1}$ then $\mathbf{a} \in Sol$. We will denote with $p : u_1 \rightsquigarrow u_2$ any path in MDD from u_1 to u_2. Also, edges between u and u' will be sometimes denoted as $e : u \to u'$. A value a of an edge $e(u, u', a)$ will be sometimes denoted as $v(e)$, while a partial assignment associated with path p will be denoted as $v(p)$. We will use $Ch[u]$ to denote the set of all outgoing (children) edges of node u. Every path corresponds to a unique assignment. Hence, the set of all solutions repre-

sented by the MDD is $Sol = \{v(p) \mid p : r \rightsquigarrow \mathbf{1}\}$. In fact, every node $u \in V_i$ can be associated with two subsets of solutions. $Sol(u) = \{v(p) \mid p : u \rightsquigarrow \mathbf{1}\} \subseteq D_i \times \ldots \times D_n$, and $Sol(r, u) = \{v(p) \mid p : r \rightsquigarrow u\} \subseteq D_1 \times \ldots \times D_{i-1}$.

Consider the MDD in Figure 1. It represents directly the solution set of T-shirt catalogue from Table 1. For each node we indicate a unique identifier u_i. All nodes in the same layer correspond to the same variable. Node u_1 is the root node. Nodes in the first, second and third layer are $V_1 = \{u_1\}$, $V_2 = \{u_2, u_3, u_4, u_5\}$ and $V_3 = \{u_6, u_7, \ldots, u_{14}\}$ respectively. For each edge e we indicate a value $v(e)$. The set of outgoing edges from, for example, node u_2 is $Ch[u_2] = \{(u_2, u_6, 0), (u_2, u_7, 1), (u_2, u_8, 2)\}$. The solution set associated with, for example, node u_3 is the set of partial assignments $Sol(u_3) = \{(1, 1), (2, 1)\}$. There are in total eleven paths from u_1 to $\mathbf{1}$, corresponding directly to the eleven products in Table 1.

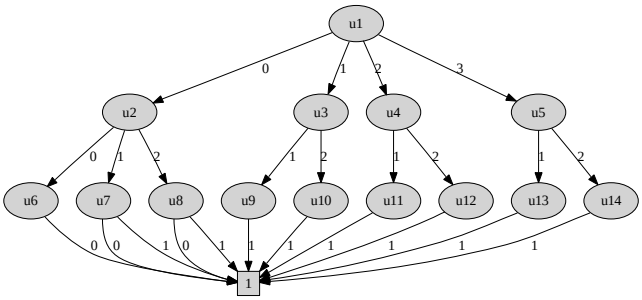
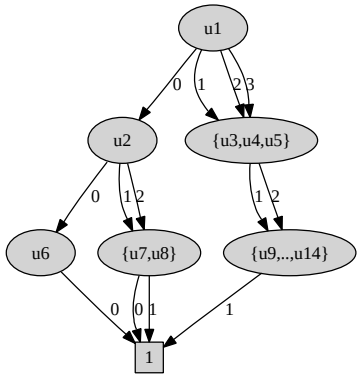


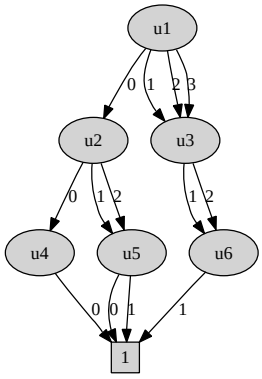
Figure 1: An MDD for the T-shirt catalogue.

The MDD from Figure 1 is as large as explicit representation in Table 1 - the number of edges is equal to the number of attribute-value pairs. However, the critical benefit of decision diagrams is that they can become exponentially smaller than the size of solution set they encode by *merging isomorphic subgraphs*. Two nodes u_1, u_2 are isomorphic if they encode the same solution set $Sol(u_1) = Sol(u_2)$. In Figure 2 we show equivalent merged MDDs for the T-shirt solution set. For the sake of clarity, we first indicate how the nodes have been merged in Figure 2(a) by using the same unique node identifiers from Figure 1. For example, nodes $\{u_3, u_4, u_5\}$ have been merged since they have the same solution sets $\{(1, 1), (2, 1)\}$. The same merged MDD, with new unique node identifiers, is shown in Figure 2(b). The utility of compressing product catalogues has already been demonstrated in [Nicholson *et al.*, 2006]. In this paper, unless emphasized otherwise, by MDD we always assume an ordered merged MDD.

On the Size and Construction of Merged MDDs. Given a variable ordering there is a unique merged MDD for a given solution set. The size of the MDD depends critically on the ordering, and could vary exponentially. The merged MDD representation of a solution set Sol with T entries defined over n attributes, $|Sol| = T$, can have at most $T \cdot n$ edges. However, data often exhibits sharing as illustrated in the T-shirt example, and a merged MDD might be exponentially



(a) A merged MDD with old unique identifiers from Figure 1 indicating how they were aggregated.



(b) A merged MDD with renamed unique node identifiers.

Figure 2: Merged MDDs for the T-Shirt example. Both graphs indicate the same structure. In Figure 2(a), for the sake of clarity, we indicate how the nodes from the uncompressed MDD in Figure 1 have been aggregated to get the MDD in Figure 2(b).

smaller. In the above example, we reduced the number of edges from 33 to 13. The effect of merging is best illustrated by considering two extreme cases. An MDD for solution set $D_1 \times \ldots \times D_n$ contains only n internal nodes and $\sum_{i=1}^{n} |D_i|$ edges while there are exponentially more solutions $T = |D_1| \cdot \ldots \cdot |D_n|$. In contrast, an MDD where every two edges at each layer are labeled with a different value is guaranteed to provide no sharing of nodes, and it would contain $T \cdot n$ nodes.

An MDD can always be constructed for a given catalogue in linear time $\mathcal{O}(T)$. We start with a direct representation, such as the T-shirt MDD in Figure 1, and in a single bottom-up pass detect and merge those nodes that root identical solution spaces. However, MDDs can be constructed also from implicit description in the form of constraint satisfaction problem. We first create an MDD M_i for each constraint c_i, and then use pairwise conjunctions to construct $M_1 \wedge \ldots \wedge M_m$. Each conjunction of two MDDs can be performed in quadratic time and space. The size of an MDD can grow exponentially in the number of variables, but in practice, for many interesting constraints the size is surprisingly small.

2.3 Functional Dependencies

Given solution set Sol defined over variables $X = \{x_1, \ldots, x_n\}$, and given solution $\mathbf{a} \in Sol$, $\mathbf{a} = (a_1, \ldots, a_n)$, we define the projection of solution $\mathbf{a}$ on variable x_i, denoted as $\mathbf{a}[x_i]$, to be the value of the i-th coordinate in the tuple, $\mathbf{a}[x_i] =_{\text{def}} a_i$. Similarly, we define projection of $\mathbf{a}$ onto a subset of variables $Y \subseteq X$, denoted as $\mathbf{a}[Y]$, as a tuple of values corresponding to variables in Y. Finally, for a given set of solutions $S \subseteq Sol$, we define the projection on subset of variables Y, denoted as $S[Y]$, as a collection of all projected tuples, $S[Y] =_{\text{def}} \{\mathbf{a}[Y] \mid \mathbf{a} \in S\}$.

For a solution set Sol, defined over variables $X = \{x_1, \ldots, x_n\}$ we say that a variable x_i is *functionally deter-*

mined by a subset of variables $Y \subseteq X$, denoted as $Y \rightarrow x_i$, if for any two solutions $\mathbf{a}_1, \mathbf{a}_2 \in Sol$, whenever $\mathbf{a}_1$ and $\mathbf{a}_2$ agree on variables Y ($\mathbf{a}_1[Y] = \mathbf{a}_2[Y]$), they also agree on variable x_i ($\mathbf{a}_1[x_i] = \mathbf{a}_2[x_i]$). Formally:

$$Y \rightarrow x_i \Leftrightarrow_{\text{def}} \forall_{\mathbf{a_1},\mathbf{a_2} \in Sol}\, \mathbf{a}_1[Y] = \mathbf{a}_2[Y] \Rightarrow \mathbf{a}_1[x_i] = \mathbf{a}_2[x_i].$$

A number of approaches are known in the database community for uncovering all minimal functional dependencies. A core operation that is executed repeatedly is testing for atomic functional dependencies of the form $Y \rightarrow x_i$. State-of-the art approaches for testing atomic dependencies first cluster data in equivalence classes with respect to the value of x_i, and then make multiple linear iterations through the dataset. This incurs linear complexity in the number of solutions $\mathcal{O}(T)$ [Huhtala *et al.*, 1999; Schlimmer, 1993]. Other approaches incur even more complexity, using sorting $\mathcal{O}(T \cdot log(T))$ or explicit comparison of all tuples $\mathcal{O}(T^2)$.

Application to Recommendation and Configuration
Detecting functional dependencies has applications in many areas. For example, uncovering that an attribute is functionally determined by values of other attributes can be of critical importance in analyzing the effect of chemical compounds on cancerogenity, studying the shopping habits of customers, etc. In this paper however, we suggest that uncovering functional dependencies can also be useful in a recommendation and interactive configuration context.

Firstly, various forms of *catalogue consistencies* can be expressed as functional dependencies. If product catalogues are frequently updated by the addition, removal or change of its items, over time various forms of inconsistencies or undesirable properties might be introduced. In particular, if care is not taken, a change of pricing policy might result in the addition of an item to the dataset that is already present in the catalogue but differs in price. Furthermore, catalogue datasets are often transformed or preprocessed for various forms of analysis or communication. Such transformations might involve the removal of "redundant" attributes, such as textual descriptions. If care is not taken, non-redundant attributes might be removed as well, thus influencing the soundness of the results of the analysis performed over the processed dataset. Thus, analyzing functional dependencies can help detect possible inconsistencies. As an illustration, checking whether "each identical configuration of a product has the same price" reduces to verifying that the price attribute is functionally determined by the remaining attributes.

Secondly, functional dependencies can help us reason about the length of user interaction with respect to the design of the user interface. If a subset of attributes $Y \subset X$ functionally determines all the other variables, $Y \rightarrow X$, then a user is guaranteed to completely specify the product as soon as all the variables Y are assigned. Hence, an interface in which a user is encouraged to first assign variables Y, might help reduce the total number of interaction steps. This could be an important addition to recent efforts towards the formal analysis of user navigation. In [Felfernig, 2006] the author used a formal model of the recommender process, based on finite state automata, to support automatic debugging of faulty models of recommender user interfaces. In [Mahmood and Ricci, 2007] the authors presented a recommender system that autonomously learns an adaptive interaction strategy, using a formal model of user interaction based on Markov decision processes. In [Hadzic and O'Sullivan, 2008] the authors introduced critique graphs as a formalism for analyzing various aspects of interaction in conversational recommender systems. In particular reachability of products through critiquing was discussed.

3 Functional Dependencies in MDDs

The main contribution of this paper is an approach to uncovering functional dependencies when a product (solution) set is compressed into a *multi-valued decision diagram* (MDD). As noted earlier, MDDs are, in the worst-case, linear in the size of the catalogue $\mathcal{O}(T)$, but they can often be exponentially smaller. Since the algorithms we suggest for uncovering functional dependencies, in this and the following sections, have quadratic complexity in the size of the MDD, whenever the MDD is sufficiently small we guarantee a sublinear-time algorithm for performing atomic dependency tests $Y \rightarrow x$.

We will first discuss how to detect *directional dependencies*, which respect variable ordering and are particularly easy to detect. We will then discuss uncovering *general dependencies* of the form $X \setminus \{x_i\} \rightarrow x_i$.

3.1 Directional Dependencies

Directional dependencies $\{x_1, \ldots, x_{i-1}\} \rightarrow x_i$ state that a variable x_i is determined by the subset of all variables preceding it in the variable ordering of the MDD. This is a particularly easy to detect class of dependencies as shown in the following proposition.

Proposition 1. *$\{x_1, \ldots, x_{i-1}\} \rightarrow x_i$ iff for all $u \in V_i$, u has only one outgoing edge $|Out(u)| = 1$.*

Proof. Let $p : r \rightsquigarrow u$ be a path from root to u, $e_1 : u \rightarrow u_1$ and $e_2 : u \rightarrow u_2$ be two outgoing edges and $p_1 : u_1 \rightsquigarrow \mathbf{1}$ and $p_2 : u_2 \rightsquigarrow \mathbf{1}$ be paths from u_1 and u_2 to terminal $\mathbf{1}$ respectively. Then paths (p, e_1, p_1) and (p, e_2, p_2) represent two solutions with identical assignments to variables $\{x_1, \ldots, x_{i-1}\}$ and two different assignments to x_i variable, $v(e_1) \neq v(e_2)$. □

Proposition 1 provides us with a simple test for checking whether $\{x_1, \ldots, x_{i-1}\} \rightarrow x_i$. It suffices that all nodes in V_i have exactly one outgoing edge. This can be easily checked by verifying that the number of nodes and outgoing edges is the same, $|V_i| = |E_i|$. The set of all variables implied by variables preceding in the order are given by:

$$Imp_{\prec} = \{x_i \mid |V_i| = |E_i|\}.$$

3.2 General Dependencies

Variables determined by subsets of variables preceding in the order, $Imp_{\prec}$, do not account for all implied variables. If variable x_i is implied by any subset $Y \subseteq X \setminus \{x_i\}$ then it will be also implied by $X \setminus \{x_i\}$. Therefore, the set of all implied variables Imp is the set of all x_i such that $X \setminus \{x_i\} \rightarrow x_i$. To detect such variables in an MDD, we will use the following proposition.

Proposition 2. *$X \setminus \{x_i\} \not\to x_i$ if and only if there is a node $u \in V_i$ with two outgoing edges $e_1 : u \to u_1$, $e_2 : u \to u_2$ such that $v(e_1) \neq v(e_2)$ and $Sol(u_1) \cap Sol(u_2) \neq \emptyset$.*

Proof. If $X \setminus \{x_i\} \not\to x_i$ then there are two solutions $\mathbf{a} = (a_1, \ldots, a_n), \mathbf{a'} = (a'_1, \ldots, a'_n)$ differing only in the i-th coordinate, $a_i \neq a'_i$. Let $p_{\mathbf{a}}$ and $p_{\mathbf{a'}}$ be the paths encoding these solutions. These paths must be of the form: $p_{\mathbf{a}} = (p, e_1, p_1)$, $p_{\mathbf{a'}}(p, e_2, p_2)$, where p is a unique path encoding $(a_1, \ldots, a_{i-1})$. Path p ends in a node $u \in V_i$. Since $a_i \neq a'_i$, $v(e_1) \neq v(e_2)$ and since $v(p_1) = v(p_2) = (a_{i+1}, \ldots, a_n)$ it follows $Sol(u_1) \cap Sol(u_2) \supseteq \{(a_{i+1}, \ldots, a_n)\}$.

On the other hand, if there is $u \in V_i$ with two outgoing edges e_1, e_2, such that $v(e_1) \neq v(e_2)$ and $Sol(u_1) \cap Sol(u_2) \neq \emptyset$ then we can choose two paths $p_1 : u_1 \rightsquigarrow \mathbf{1}$, $p_2 : u_2 \rightsquigarrow \mathbf{1}$ such that $v(p_1) = v(p_2)$. It suffices to choose any path from root to u, $p : r \rightsquigarrow u$ to construct paths $p_{\mathbf{a}} = (p, e_1, p_1)$, $p_{\mathbf{a'}}(p, e_2, p_2)$ which encode solutions differing only at the i-th coordinate and thus proving $X \setminus \{x_i\} \not\to x_i$. □

Assume that for each pair of nodes in the same layer u_1, u_2 we have precomputed Boolean indicators $D[u_1, u_2]$

$$D[u_1, u_2] = 1 \Leftrightarrow Sol(u_1) \cap Sol(u_2) \neq \emptyset.$$

Whenever we encounter a pair of nodes (u_1, u_2) such that $D[u_1, u_2] = 1$, we are guaranteed that there are at least two paths $p_1 : u_1 \rightsquigarrow \mathbf{1}$ and $p_2 : u_2 \rightsquigarrow \mathbf{1}$ encoding the same solution $v(p_1) = v(p_2)$. Given such labels, we can compute all functionally determined variables using Algorithm 1. In each layer we check for all pairs of edges with the same parent $e_1(u, u_1, a_1)$,$e_2(u, u_2, a_2)$ whether $D[u_1, u_2] = 1$. As soon as such edges are found we have proven that x_i is not implied and we may proceed to the next layer. The algorithm runs in $\mathcal{O}(\sum_{i=1}^{n} |V_i| \cdot |D_i|^2)$ steps, since for each node in each layer $u \in V_i$, we compare in worst case all pairs of its children edges, and there are at most $|D_i| \times (|D_i| - 1)/2$ such pairs. The space complexity is $\mathcal{O}(\sum_{i=1}^{n} |V_i|^2)$ since we have to store Boolean indicators $D[u_1, u_2]$ for each pair $(u_1, u_2) \in V_i^2$.

Algorithm 1: Compute functionally determined variables.

```
Data: MDD M(V, E)
Imp = X;
foreach i = 1, ..., n do
    foreach u ∈ V_i, |Ch(u)| > 1 do
        foreach e_1 : u → u_1, e_2 : u → u_2 do
            if v(e_1) ≠ v(e_2) ∧ D[u_1, u_2] = 1 then
                Imp ← Imp \ {x_i};
                go to next layer;
return Imp;
```

Compatibility pairs $D[u_1, u_2]$ can be computed in quadratic time and space using a dynamic programming scheme from Algorithm 2. We first initialize $D[u_1, u_2] = 0$ for all pairs of nodes, except for the terminal $\mathbf{1}$, setting $D[\mathbf{1}, \mathbf{1}] = 1$. We then, in a bottom-up manner, traverse the MDD. The recursive relationship used for dynamic programming is based on observing that $D[u_1, u_2] = 1$ iff there are two outgoing edges $e_1 : u_1 \to u'_1, e_2 : u_2 \to u'_2$ such that $v(e_1) = v(e_2) \wedge D[u'_1, u'_2] = 1$. The algorithm runs in $\mathcal{O}(\sum_{i=1}^{n} |E_i|^2)$ time since in each layer E_i each pair of edges is compared at most once. The algorithm takes $\Theta(\sum_i |V_i|^2)$ space, since we introduce a compatibility indicator for each pair of nodes in the layer.

Algorithm 2: Compute Boolean indicators.

```
Data: MDD M(V, E)
D[·, ·] = 0, D[1, 1] = 1;
foreach i = n, ..., 1 do
    foreach (u_1, u_2) ∈ V_i × V_i do
        if u_1 = u_2 then
            D[u_1, u_2] = 1;
            continue;
        foreach e_1 : u_1 → u'_1, e_2 : u_2 → u'_2 do
            if v(e_1) = v(e_2) ∧ D[u'_1, u'_2] = 1 then
                D[u_1, u_2] = 1;
                break;
return D;
```

4 Subset-Induced Dependencies

Given a subset of variables $Y \subseteq X$ we may want to compute the set of all variables implied by Y:

$$Imp_Y = \{x_i \in X \setminus Y \mid Y \to x_i\}.$$

This could help us to evaluate an overall impact of a subset of variables. We could exploit this information in a number of different settings. In particular, if we find Y such that $Imp_Y = X \setminus Y$, then regardless of how we assign variables Y, we would completely specify entire solution. This could be important for increasing the usability of user interaction since, if a user is assigning only variables Y we guarantee that the number of user interactions before completely specifying a solution is at most $|Y|$. Furthermore, such an information could help organize the visual layout of the variables in a user interface. If a variable x_i is determined by variables Y, by displaying x_i closer to Y in the user interface, a user would faster evaluate implications of his assignments to Y variables.

By definition, a variable x_i is not implied by Y iff there are two solutions $\mathbf{a_1}, \mathbf{a_2}$ such that $\mathbf{a_1}[Y] = \mathbf{a_2}[Y]$ and $a_i \neq a'_i$. Recall that $S[Y]$ denotes a projection of set S on variables Y. An observation that would help us detect such variables is provided in the following proposition.

Proposition 3. *$Y \not\to x_i$ iff there are two edges $e_1, e_2 \in E_i$, $e_1 : u_1 \to u'_1$, $e_2 : u_2 \to u'_2$, such that $v(e_1) \neq v(e_2)$ and $Sol(r, u_1)[Y] \cap Sol(r, u_2)[Y] \neq \emptyset$ and $Sol(u'_1)[Y] \cap Sol(u'_2)[Y] \neq \emptyset$.*

Assuming that for every pair of nodes u_1, u_2 we have computed Boolean indicators:

$$U_Y[u_1, u_2] = 1 \Leftrightarrow Sol(r, u_1)[Y] \cap Sol(r, u_2)[Y] \neq \emptyset$$

$$D_Y[u_1, u_2] = 1 \Leftrightarrow Sol(u_1)[Y] \cap Sol(u_2)[Y] \neq \emptyset$$

we could use an adaptation of Algorithm 1 to detect all variables implied by subset Y. The adaptation is presented in Algorithm 3.

Algorithm 3: Compute Y-implied variables.

Data: MDD $M(V, E), Y \subset X, U_Y, D_Y$
$Imp_Y = X$;
foreach $i = 1, \ldots, n$ **do**
  foreach $(u_1, u_2) \in V_i \times V_i$ **do**
    foreach $e_1 : u_1 \to u_1', e_2 : u_2 \to u_2'$ **do**
      if $v(e_1) \neq v(e_2) \wedge U_Y[u_1, u_2] = 1 \wedge D_Y[u_1', u_2'] = 1$ **then**
        $Imp_Y \leftarrow Imp_Y \setminus \{x_i\}$;
        go to next layer;
return Imp_Y;

Compatibility labels U_Y, D_Y can be computed in quadratic time and space using an adaptation of Algorithm 2 which is presented in Algorithm 4. To construct a recursive relationship on which the computation is based, for a given pair of nodes $u_1, u_2 \in V_j$ we have to differentiate between two cases. If $x_j \notin Y$, then $D_Y[u_1, u_2] = 1$ iff there are two outgoing edges $e_1 : u_1 \to u_1', e_2 : u_2 \to u_2'$ such that $D_Y[u_1', u_2'] = 1$ regardless of whether $v(e_1) = v(e_2)$ or $v(e_1) \neq v(e_2)$. If $x_j \in Y$ then $D_Y[u_1, u_2] = 1$ iff in addition to $D_Y[u_1', u_2'] = 1$ it also holds $v(e_1) = v(e_2)$. Compatibility labels U_Y are computed in an analogous manner.

The algorithm runs in $\mathcal{O}(\sum_{i=1}^{n} |E_i|^2)$ time since for both traversals, in each layer E_i each pair of edges is compared at most once. The algorithm takes $\Theta(\sum_i |V_i|^2)$ space, since we introduce two Boolean compatibility indicators U_Y, D_Y for each pair of nodes in the layer.

Algorithm 4: Compute Y-Boolean indicators.

Data: MDD $M(V, E), Y \subset X$
$D_Y[\cdot, \cdot] = 0, D_Y[\mathbf{1}, \mathbf{1}] = 1$;
foreach $i = n, \ldots, 1$ **do**
  foreach $(u_1, u_2) \in V_i \times V_i$ **do**
    if $u_1 = u_2$ **then**
      $D_Y[u_1, u_2] = 1$;
      continue;
    foreach $e_1 : u_1 \to u_1', e_2 : u_2 \to u_2'$ **do**
      if $x_i \notin Y \wedge D_Y[u_1', u_2'] = 1$ **then**
        $D_Y[u_1, u_2] = 1$;
        break;
      else if $x_i \in Y \wedge v(e_1) = v(e_2) \wedge D_Y[u_1', u_2'] = 1$ **then**
        $D_Y[u_1, u_2] = 1$;
        break;
$U_Y[\cdot, \cdot] = 0, U_Y[r, r] = 1$;
foreach $i = 1, \ldots, n$ **do**
  foreach $(u_1', u_2') \in V_{i+1} \times V_{i+1}$ **do**
    if $u_1' = u_2'$ **then**
      $U_Y[u_1', u_2'] = 1$;
      continue;
    foreach $e_1 : u_1 \to u_1', e_2 : u_2 \to u_2'$ **do**
      if $x_i \notin Y \wedge U_Y[u_1, u_2] = 1$ **then**
        $U_Y[u_1', u_2'] = 1$;
        break;
      else if $x_i \in Y \wedge v(e_1) = v(e_2) \wedge U_Y[u_1, u_2] = 1$ **then**
        $U_Y[u_1', u_2'] = 1$;
        break;

4.1 Finding Minimal Dependencies

Given a subset of variables $Y_0 \subseteq X$ such that $X \setminus Y_0 \to Y_0$, it is often required to compute the set of all *minimal* subsets of variables $Y \subseteq X \setminus Y_0$ that imply Y_0. In other words, we want to compute:

$$\mathcal{Y} = \{Y \subseteq X \setminus Y_0 \mid Y \to Y_0,\ s.t.\ \nexists_{Y' \subset Y} Y' \to Y_0\}.$$

There could be an exponential number of such sets, and a number of approaches has been developed that operate on a set-containment lattice [Huhtala *et al.*, 1999] and avoid unnecessary tests $Y \to x$. For example, whenever Y determines x, $Y \to x$, all supersets of Y also determine x. A number of similar optimizations are implemented in [Huhtala *et al.*, 1999]. Extending existing approaches by incorporating our MDD-tests is an interesting direction for future research, but falls out of the scope of this paper.

5 Approximative Dependencies

We have so far discussed only *exact* dependencies, i.e. we detect only whether variable x_i is determined or not. However, a subset of variables Y might have a significant implicative influence on a variable x even though it does not imply it exactly. Therefore we are interested in computing a *degree of dependency* $Y \to x$. Let $d(Y, x) \in [0..1]$ denote such a degree. There is a number of ways to define it. In [Huhtala *et al.*, 1999] the authors use for the measure a minimal percentage of solutions that has to be removed in order for dependency to hold. We however use the following definition:

$$d(Y, x) = \frac{|Sol[Y]|}{|Sol[Y \cup \{x\}]|}$$

When $Y \to x$, then $|Sol[Y]| = |Sol[Y \cup \{x\}]|$, and $d(Y, x) = 1$. On the other hand, when x is completely undetermined by Y, then every assignment to Y variables $\mathbf{a} \in Sol[Y]$ can be combined with all values in domain for x, D_x. In that case,

$$d(Y, x) = \frac{1}{|D_x|}.$$

Hence, to each dependency statement $Y \to x$ we assign a measure

$$d(Y, x) \in [\frac{1}{|D_x|}, 1]$$

where larger values indicate larger degrees of dependency, and when $d(Y, x) = 1$, $Y \rightarrow x$ holds exactly. We can interpret this measure as follows - whenever $d(Y, x) = d$, every assignment to variables Y can on average be combined with $\frac{1}{d}$ values of x.

Note that such a statistic can be easily computed using a BDD-based representation of solution set Sol. A BDD-package *BuDDy* [Lind-Nielsen, online] supports both projection and counting operations. Counting the number of solutions in a BDD is an efficient operation - linear in the number of nodes. While computing a BDD representation of a projected solution space, such as $Sol[Y]$, can in theory increase the size of the BDD - in practice projecting out variables is almost always an efficient operation that decreases the number of nodes significantly.

6 Experimental Evaluation

We have applied our techniques to four well-known product catalogues that are frequently used in recommender system research. These are related to digital cameras, laptop computers, property lettings and travel [Nicholson *et al.*, 2006]. For the purposes of our evaluation, we analyzed the data under the same adjustments that are usually done for experimental evaluation. Firstly, all unique identifiers or textual descriptions, were removed. Secondly, all declared domain values that did not appear in at least one product, but appeared in the specification, were also removed. If some values appeared in datasets but were not declared, we added them to model specification.

A summary of the properties of the instances are reported in Table 2. For each instance, *Cameras, Laptops, Travel, Lettings* we show the number of rows in the initial explicit solution set representation, the number of solutions Sol extracted from the MDD representation, the number of variables X, and minimal, maximal and average domain size. We uncovered that three out of four datasets contain duplicate entries, since the number of solutions is smaller than the number of rows.

Table 2: Basic properties of product catalogues. For each instance we show the number of rows in the initial table representation, number of solutions Sol extracted from the MDD representation, number of variables X, and minimal, maximal and average domain size.

Instance	*Rows*	$\lvert Sol\rvert$	$\mathbf{X}$	$\mathbf{d_{min}}$	$\mathbf{d_{max}}$	$\mathbf{d_{avg}}$
Cameras	210	210	9	5	165	40
Laptops	693	683	14	2	438	42
Travel	1470	1461	7	4	839	134
Lettings	794	751	6	2	174	45

Dependency Analysis

After compiling catalogues into MDDs, we performed a dependency analysis $X \setminus \{x_i\} \rightarrow x_i$ for each instance and each attribute. A detailed analysis of product catalogues is presented in Table 3. For each instance, we list all variables X and for each variable $x_i \in X$ we indicate its *type* (Categorical, Numerical, Boolean), *domain size* and *degree of dependence*:

$$d(X \setminus \{x_i\}, x_i) = \frac{|Sol[X \setminus \{x_i\}]|}{|Sol]|}.$$

Recall that the degree of dependence $d(Y, x)$ captures how many values of x_i can on average be combined with every assignment to variables Y. For example, if $d(Y, x) = 1/2$, then every assignment to variables Y can on average be combined with 2 values in a domain of x. If $d(Y, x) = 1$ then for every assignment to Y variables there is exactly one compatible value in domain of x and hence, x is functionally determined by Y. We can see from the table that almost all variables are functionally determined, and those that are not have a very high level of functional dependency. This is not surprising given that the price of the product is the most distinguishing attribute - behaving similarly as a unique key in a database. However, to our surprise, price is *not functionally dependent* in any of the catalogues! This indicates that in each catalogue there are identical configurations which have different prices! This is particularly emphasized for the *Lettings* catalogue, where degree of dependence is 0.563. This means that each configuration of remaining attributes on average has two different prices.

After analyzing more closely the variable specification and original and processed datasets, we discovered that a *Street* attribute of the *Lettings* catalogue was not declared in the dataset specification. Therefore, the preprocessing step ignored the corresponding column. Hence, a number of lettings with identical specifications and in the same region were treated as identical even though they were offered in different streets of the same region. For the same reason, the number of duplicate entries in Table 2 of the *Lettings* catalogue was disproportionably large.

7 Conclusions

In this paper we presented an approach to computing functional dependencies over an MDD representation of a product catalogue. We focused on verifying catalogue consistencies as an application relevant within an online configuration/recommendation context. Using functional dependencies as an analytic tool we discovered that a set of publicly available product catalogues exhibits specific characteristics that have not been noted to-date; some of these characteristics can be regarded as bugs in the catalogue definition or in the preprocessing step of the catalogues. This warrants caution in the future investigations in the area of web-based configuration and recommender systems that rely on preprocessed forms of product catalogues. The fact that the datasets might violate some consistency criteria should be taken into account when drawing conclusions from experimental evaluations.

We also identified that functional dependencies can be used to formally reason about the length of user interaction. This topic fits within recent research about recommender systems [Felfernig, 2006; Mahmood and Ricci, 2007; Hadzic and O'Sullivan, 2008] but was not the focus of this paper. Instead, it would be pursued in future work. In addition, in the future we plan to further investigate the utility

Table 3: A dependency analysis of product catalogues *Cameras, Laptops, Travel* and *Lettings*. For each instance, we list variables X and for each variable we indicate its *type* , *domain size* and *degree of dependence*. Variable type *Boolean** indicates that beside *yes* or *no* values, a value *unknown* is also permitted.

Instance	Variable	Type	Domain size	Degree of dependence
Cameras	Format	Categorical	5	1.0
	Storage type	Categorical	7	1.0
	Storage amount	Numerical	12	1.0
	Manufacturer	Categorical	15	1.0
	Optical zoom	Numerical	21	1.0
	Digital zoom	Numerical	22	1.0
	Resolution	Numerical	31	1.0
	Weight	Numerical	87	1.0
	Price	Numerical	165	0.966
Laptops	Microphone	Boolean	2	0.995
	Speakers	Boolean	2	0.998
	DVD	Boolean*	2	1.0
	Floppy	Boolean	2	1.0
	Modem	Boolean*	3	1.0
	CDROM	Boolean*	3	1.0
	Pointing device	Categorical	3	1.0
	RAM	Numerical	9	1.0
	Screen size	Numerical	9	1.0
	Processor type	Categorical	10	1.0
	Processor speed	Numerical	15	0.998
	HD size	Numerical	28	0.998
	Weight	Numerical	64	1.0
	Price	Numerical	438	0.855
Travel	Transport	Categorical	4	0.999
	Accommodation	Categorical	6	0.998
	Holiday type	Categorical	8	0.964
	Duration	Numerical	9	0.998
	Number of persons	Numerical	11	0.991
	Region	Categorical	65	0.971
	Price	Numerical	839	0.809
Lettings	Type	Categorical	2	0.993
	Furnished	Boolean	2	0.968
	Baths	Numerical	6	0.981
	Beds	Numerical	8	0.955
	Area	Categorical	80	0.650
	Price	Numerical	174	0.563

of decision diagrams for recommending general configurable products.

Acknowledgements

Hadzic is supported by a Post-doctoral Research Fellowship from the Irish Research Council for Science, Engineering and Technology. O'Sullivan is supported by Science Foundation Ireland (Grant Number 05/IN/I886). We would like to thank anonymous reviewers for their useful comments.

References

[Bryant, 1986] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 1986.

[Drechsler, 2001] Rolf Drechsler. Binary decision diagrams in theory and practice. *International Journal on Software Tools for Technology Transfer (STTT)*, 3(2):112–136, May 2001.

[Felfernig, 2006] Alexander Felfernig. Diagnosing faulty transitions in recommender user interface descriptions. In *Advances in Applied Artificial Intelligence*, pages 869–878. Springer-Verlag, 2006.

[Hadzic and O'Sullivan, 2008] Tarik Hadzic and Barry O'Sullivan. Critique graphs for catalogue navigation. In *RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems*, pages 115–122, New York, NY, USA, 2008. ACM.

[Huhtala *et al.*, 1999] Ykä Huhtala, Juha Kärkkäinen, Pasi Porkka, and Hannu Toivonen. Tane: An efficient algorithm for discovering functional and approximate dependencies. *The Computer Journal*, 42(2):100–111, March 1999.

[Lind-Nielsen, online] J. Lind-Nielsen. BuDDy - A Binary Decision Diagram Package. `http://sourceforge.net/projects/buddy`, online.

[Mahmood and Ricci, 2007] Tariq Mahmood and Francesco Ricci. Learning and adaptivity in interactive recommender systems. In *ICEC '07: Proceedings of the ninth international conference on Electronic commerce*, pages 75–84, New York, NY, USA, 2007. ACM.

[Meinel and Theobald, 1998] C. Meinel and T. Theobald. *Algorithms and Data Structures in VLSI Design*. Springer, 1998.

[Nicholson *et al.*, 2006] Ross Nicholson, Derek Bridge, and Nic Wilson. Decision diagrams: Fast and flexible support for case retrieval and recommendation. In *Proceedings of Eighth European Conference on Case-Based Reasoning (ECCBR 2006)*, 2006.

[Schlimmer, 1993] Jeffrey C. Schlimmer. Efficiently inducing determinations: A complete and systematic search algorithm that uses optimal pruning. In *ICML*, pages 284–290, 1993.

[Wegener, 2000] Ingo Wegener. *Branching Programs and Binary Decision Diagrams*. Society for Industrial and Applied Mathematics (SIAM), 2000.

Effectiveness of different recommender algorithms in the Mobile Internet: A case study

Kolja Hegelich and Dietmar Jannach
Technische Universität Dortmund
44221 Dortmund, Germany
kolja.hegelich@tu-dortmund.de, dietmar.jannach@tu-dortmund.de

Abstract

Despite the broad use of Recommender Systems (RS) technology in various domains, the number of publicly available reports on the actual business value of such systems is limited.

This paper presents first results of an empirical evaluation of how different recommendation algorithms affect the navigation and buying behavior of a sample of over 155.000 different customers on a commercial Mobile Internet portal for cell phone games. The evaluated RS algorithms include item-based collaborative filtering, SlopeOne, a content-based as well as a hybrid technique, which were compared with naive approaches based on top-selling and top-rated items.

The analysis shows that RS measurably affected the navigation and buying behavior of the portal visitors. The personalized recommendation lists not only attracted more clicks on detailed item descriptions but also lead to an overall sales increase when compared with control groups that received non-personalized recommendations or no recommendations during the evaluation period.

The comparison of different algorithms brought no clear winner that consistently outperformed the others. However, the results indicate that the choice of the recommendation technique should depend on the specific navigational situation in which recommendation lists are presented.

Introduction & previous studies

Although the interest in recommender systems technology has been increasing in the last years both in industry and research and although recommender applications can nowadays be found on many web sites of online retailers, nearly no studies about the actual business value of such systems have been published that are based on real-world transaction data.

In the research community, the performance of a recommender system is mainly measured based on its *accuracy* with respect to predicting whether a user will like a certain item or not[1]. The implicit assumption is that the online user – after establishing trust in the system's recommendations or because of curiosity – will more often buy these recommended items from the shop.

However, a shop owner's key performance indicators related to a personalized web application such as a recommender system are different ones. Establishing a trustful customer relationship, providing extra service to customers by proposing interesting items, maintaining good recommendation accuracy and so on are only a means to an end. While these aspects are undoubtedly important for the long-term success of a business, for an online retailer, the important performance indicators are related (a) to the increase of the *conversion rate*, i.e., how web site visitors can be turned into buyers, and (b) to questions of how to influence the visitors in a way that they buy more or more profitable items.

Unfortunately, only few real-world studies in that context are available because large online retailers do not publish their evaluations of the business value of recommender systems. Only a few exceptions exist. Dias et al. (Dias et al. 2008), for instance, recently presented the results of a 21-month evaluation of their probabilistic item-based recommender system running on a large Swiss e-grocer web portal. Their measures include "shopper penetration", "direct extra revenue" and "indirect extra revenue". Their analysis showed different interesting points. First, a relatively small (when compared to overall sales) extra revenue can be generated directly by the recommender. The fact that direct revenues measurably increased when the probabilistic model went through a periodic update suggests that good recommendation *accuracy* is still important, despite some legitimate criticism of simple accuracy measures (McNee, Riedl, and Konstan 2006). The more important business value, however, comes from *indirect* revenues caused by the recommender systems. Indirect revenues include the money spent on repeated purchases of items initially recommended by the system and on items sold from categories to which the customer was newly introduced to through a recommended item. This in turn also supports the theory that *diversity* in recommendation lists is a valuable property as "unexpected" items in these lists may help to direct users to other, possibly interesting categories.

An earlier evaluation based on real-world data was presented in (Shani, Brafman, and Heckerman 2002), where the authors performed different experiments on an online bookstore. During their experiment, visitors of the web shop re-

[1] See (Herlocker et al. 2004) for an overview on evaluation metrics for recommender systems.

ceived buying proposals either from a "predictive" or a new Markov Decision Process recommender. Thus, they were able to compare the respective *profits* that were generated by different techniques during the observation period. In addition, at least for a period of seven days, the recommendation functionality was fully removed from the web shop. Although this sample is statistically too small, the comparison of sales numbers of two consecutive weeks (one with and one without the recommender) showed a 17% drop in the recommender-less week.

Another initial study on how recommender systems influences the buying behavior of web shop visitors is presented in (Zanker et al. 2006). In this work, it was shown that the recommendations of a virtual advisor for premium cigars can stimulate visitors to buy cigars other than the well-known *Cohibas* and thus increase *sales diversity*, which is interesting from an up-selling and cross-selling perspective and could also create "indirect revenue" as described in (Dias et al. 2008); see also (Fleder and Hosanagar 2007) for a discussion of the role of sales diversity in recommender systems.

In (Zanker et al. 2008), a different study using the same recommendation technology was made in the tourism industry, where it could be observed that the number of accommodation availability enquiries is measurably higher when web site visitors are guided by the virtual advisor. Another evaluation of how different information types and recommendation "sources" influence consumers can be found in (Senecal and Nantel 2004).

Similar to these works, our paper focuses on evaluating the business value of recommender systems in a commercial context. In addition, it aims to answer the question whether certain algorithms perform better than others in a certain environment and application domain in the line of the work of, e.g., (Breese, Heckerman, and Kadie 1998) or (Zanker et al. 2007).

Application and personalization overview

The study presented in this paper was conducted in the context of a Mobile Internet portal of a large telecommunications provider in Germany. Customers access this portal through their mobile devices and are offered a wide range of applications and games, which they can directly purchase and download to their cell phones.

Figure 1 shows the entry screen of the games area of the portal. Customers explore the item catalog in the following ways:

- Through *manually-edited or non-personalized lists* such as "New items" or "Top10 items" (top area of screen).
- Through direct text or image links (teasers) to certain items that are shown on the middle area of the start screen.
- Through predefined *standard categories* (lower area) such as "A - Z", "From 99 Cent", or "Action & Shooters".
- In addition, after a purchase, when the payment confirmation is displayed, customers are presented with a list of other, possibly interesting items (post-sales recommendation).

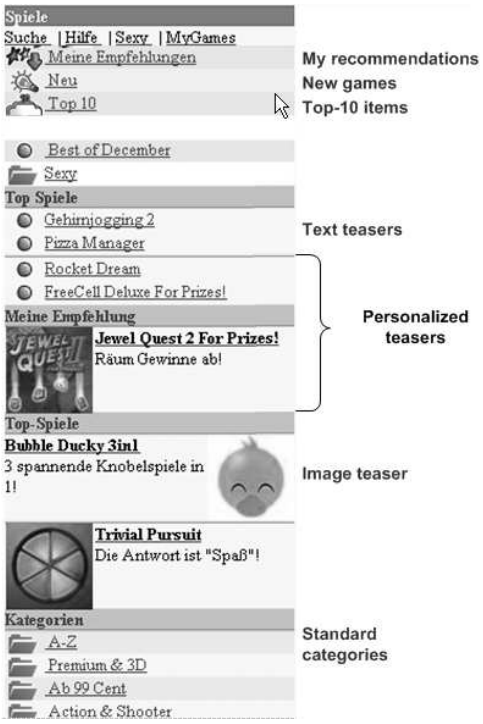


Figure 1: Catalog navigation and categories

Accordingly, the portal was extended with personalized content as follows.

1. A new top-level link "My Recommendations" was introduced that leads to a personalized recommendation list. ("Meine Empfehlungen" in German).
2. The games presented in the lower two of the four text teasers and the first image teaser on the start page were personalized. Due to existing contracts the first two text links and the two lower image links were manually predefined. The manually edited links remained the same during the whole experiments, which allowed us to analyze the effects of personalizing the other links independently.
3. The lists in the standard categories such as "99 Cent" were personalized except for categories such as "A-Z", which have a "natural" ordering.
4. The games presented on the post-sales page were also personalized.

During the experiments, different algorithms were used to calculate the personalized recommendations. In order to measure the effect of personalization, members of the control group were shown non-personalized or manually-edited lists which are based on the release date of the game.

Customers can immediately purchase and download games through the portal by choosing items from the presented lists. The relation between their navigation and buying behavior can therefore be easily determined as all portal visitors are always logged-in. Several thousand games (across all categories) are downloaded each day through the

platform. The prices for the games range from free evaluation versions (demos) over "99Cent-Games" to a few Euro for premium games and the amounts are directly charged to the customer's monthly invoice. Note that in contrast to the study in (Dias et al. 2008), where users purchase the same goods repeatedly, customers in our domain only purchase the same item once, i.e., our domain is similar to popular recommender systems application areas such as books and movies.

From the perspective of the application domain, the presented game portal stands in the line of previous works in the area of recommender systems for mobile users. Recent works in the field of mobile recommenders include, e.g., (Miller et al. 2003), (Cho, Kim, and Kim 2004), (van der Heijden, Kotsis, and Kronsteiner 2005), (Ricci and Nguyen 2007), (Li, Wang, and Geng 2008) or (Nguyen and Ricci 2008). Content personalization approaches for the Mobile Internet are presented also in (Pazzani 2002),(Billsus and Pazzani 2007) and (Smyth, Cotter, and Oman 2007). In (Smyth and Cotter 2002), finally, the effects of personalizing the navigational structure on a commercial WAP portal are reported.

It can be expected that this area will attract even more attention in the future because of the rapid developments in the hardware sector and the increasing availability of cheap and fast mobile Internet connections. Note that in contrast to some other approaches, our system does not exploit additionally available information such as the current geographical position or demographic and other customer information known to the service provider. Standard limitations of Mobile Internet applications such as relatively small network capacity and limited display sizes however apply.

Algorithms and ratings

During the four week evaluation period, customers were assigned to seven different groups when they entered the games section of the portal. For each group, the item lists were generated in a different way. For the first four groups, the following recommendation algorithms were used.

- Item-based collaborative filtering (CF) (Sarwar et al. 2001) as also used by Amazon.com (Linden, Smith, and York 2003).
- The recent and comparably simple SlopeOne algorithm (Lemire and Maclachlan 2005).
- A content-based method using a TF-IDF representation of the item descriptions and the cosine similarity measure.
- A "switching" (Burke 2002) hybrid algorithm that uses the content-based method when less than 8 item ratings are available and item-based collaborative filtering otherwise.

Two groups received non-personalized item lists, one based on the average item rating ("TopRating") and one based on the sales numbers (top sellers). For the final group, the control group, the recommendation lists were manually edited as it was before the personalization features have been introduced. Within most categories, the ordering was based on the release date of the game or chosen based on existing contracts. The top-level link "My Recommendations" was not available for the control group. During the whole evaluation period, customers remained in their originally assigned group.

From all customers that visited the games portal during the evaluation, a representative sample of over 155.000 was included in the experiment, so each group consisted of around 22.300 customers. Note that only such customers were chosen for which all algorithms were able to produce a recommendation, i.e., users for which a minimum number of ratings already existed. Also only such frequent customers were assigned to the control group and the groups receiving non-personalized recommendations, which guarantees that similar customer segments are compared. The catalog of recommendable items consisted of about 1.000 games.

A five-point rating scale from -2 to $+2$ was used in the experiments. Since the number of explicit item ratings was very low and only about two percent of the customers have issued at least one rating, also implicit ratings have been taken into account: both clicks on item details as well as actual purchases were interpreted as implicit ratings. When no explicit rating was given, a view on item details was interpreted as a rating of 0 (medium); several clicks on the same item were not counted. An actual purchase was interpreted as a rating of 1 (good) for the item. Explicit ratings override these implicit ratings.

In order to achieve the best possible recommendation accuracy, the item similarities and the average differences for the collaborative filtering and the SlopeOne techniques were computed using the full customer base and not only the 155.000 customer sub-sample.

Evaluation

The following hypotheses are in the center of our evaluation.

- H1: Personalized recommendations attract more customers to detailed product information pages (item view conversion rate).
- H2: Personalized recommendations help to turn more visitors into buyers (sales conversion rate).
- H3: Personalized recommendations stimulate individual customers to view more items.
- H4: Personalized recommendations stimulate individual customers to buy more items.

The detailed evaluation will show that depending on the navigational situation of the portal visitor different phenomena with respect to the effectiveness of recommendation algorithms can be observed. Before considering the overall effect of the use of recommendation technology on the portal, we will discuss the individual results obtained for these different situations.

Measurement 1: My Recommendations

The following results are related to the personalized recommendation list that is presented when the customer clicks on the "My Recommendations" link as shown in the top area of Figure 1. Throughout the evaluation, we will use different

colors to highlight data rows in the charts that are significantly different ($p < 0.01$) from each other.

The conversion rate measurements (hypotheses H1 and H2) are given in Figure 2, which depicts the *item view conversion rate* for visitors of the "My Recommendations" list, and Figure 3 that shows how many of the users that visited the "My Recommendations" section actually purchased an item[2].

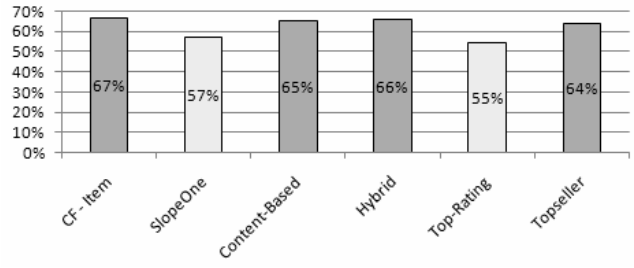


Figure 2: Conversion rate: Item views to "My Recommendations" visits

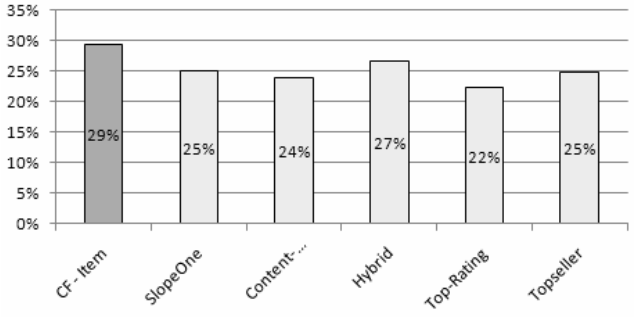


Figure 3: Conversion rate: Buyers to My Recommendations visits

In Figure 2 we see that the different algorithms fall into two groups: One, where about two thirds of of the customers actually click on at least one of the presented items and one, where only 55% are interested in the recommended items. Considering the actual numbers, the differences between the two groups are significant ($p < 0.01$).

From the personalized methods, only the SlopeOne algorithm did not attract significantly more visitors than the non-personalized list of top rated items. Interestingly, the non-personalized top seller list also has a good item view conversion rate, i.e., placing generally-liked, top-selling items in a recommendation list seems to work quite well in the domain.

When the sales conversion rate is considered, we see in Figure 3 that only the CF method helps to turn more visitors into buyers (Hypothesis H2).

[2]In Figures 2 to 5, the control group is not depicted, because the "My Recommendations" section, which was newly introduced for measuring the impact of personalization, was not available for them.

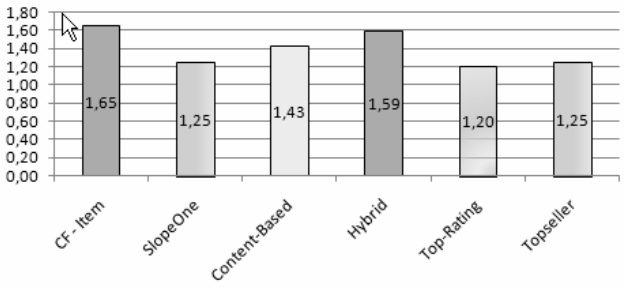


Figure 4: Item views per "My Recommendations" visits

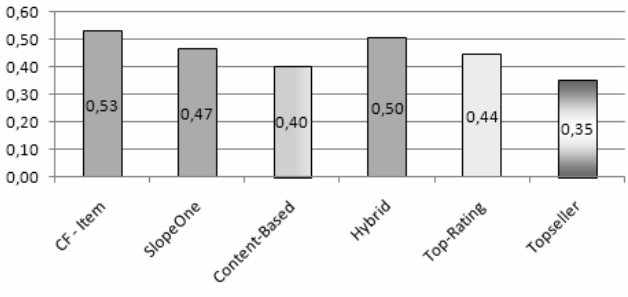


Figure 5: Item purchases to "My Recommendations" visits

The evidence for our hypotheses H3 (more item views per customer) and H4 (more purchases per customer) in the context of the "My Recommendations" section can be seen in Figures 4 and 5. Figure 4 shows that all recommendation algorithms (except for SlopeOne) stimulate users to click on more items. Compared with the findings with respect to the conversion rates, this can be interpreted as follows: while top seller lists help to stimulate one or the other customer to click on an item detail, personalized lists seem to contain more items that are interesting to a customer.

When it comes to actual purchases (game downloads), Figure 5 shows that most personalized methods and even the simple SlopeOne algorithm outperform the non-personalized approaches.

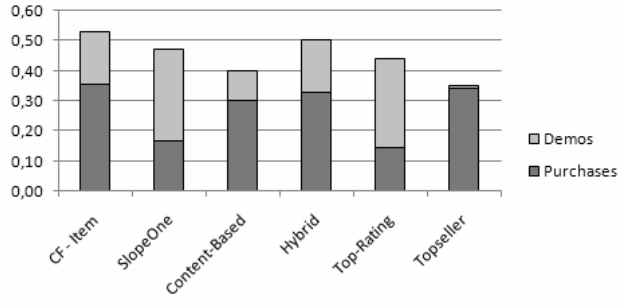


Figure 6: Game purchases and demo downloads in "My Recommendations"

Note that for some of the games provided on the mobile

portal, free evaluation versions (demos) are available. If not mentioned otherwise, all numbers given with respect to conversion rates and sales figures are related to *all item downloads*, i.e. free demos plus actual game purchases. Figure 6 now repeats the numbers of Figure 5, but also shows the fraction of demo downloads and purchased games. Due to the nature of the algorithms and the particularities of the application (see more details in Measurement 4) , the recommendation lists produced by the TopRating and SlopeOne methods contain a relatively high portion of demo games. Given the high number of actual downloads, these demo recommendations seem to be well-accepted, but unfortunately, these two techniques perform particularly poor when the games are not free. The item-based, content-based and hybrid technique, on the other hand, not only help to sell as many items as a simple top-seller promotion but also make users curios about demo games. The TopRating method only raises interest in demo versions. The list of top selling items is generally dominated by non-free, mainstream games, which explains the fact nearly no demo games are chosen by the users.

Measurement 2: Post-sales recommendations

The next navigational situation in which product recommendations are made is when a customer has purchased an item and the payment receipt is displayed. About 90.000 customers who actually bought at least one item during the evaluation period have been involved in the experiment. Overall, the evaluation sample contains more than 230.000 views of the post-sales 5-item recommendation lists, meaning that on average, customers bought more than one item.

The experimental setup is nearly identical with Measurement 1 and customers received their recommendations based on different recommendation algorithms. The recommendation list of the control group was manually edited and ordered by game release date. Items that the current customer has already purchased before were removed from these lists.

The same hypotheses were tested in this experiment, i.e., to what extent recommender systems stimulate customers to view and buy more items. The results are shown in Figures 7 to 10.

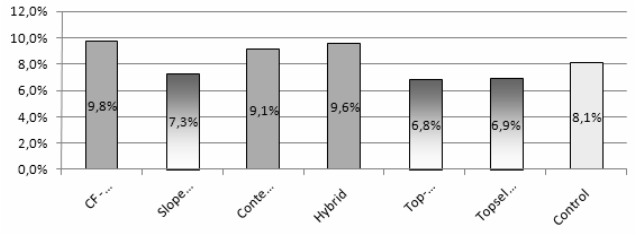


Figure 7: Conversion rate: Item views to post-sales list views

With respect to the conversion rates, the following observations can be made. First, the manually edited list of recent items (viewed by the control group) worked quite well and has raised more customer interest than the non-personalized techniques and even the SlopeOne algorithm

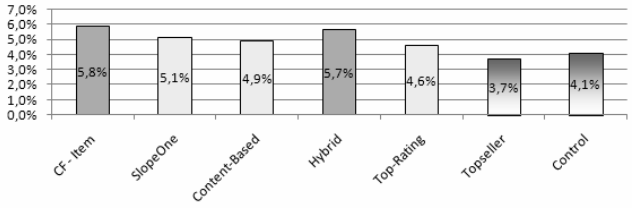


Figure 8: Conversion rate: Buyers to post-sales list views

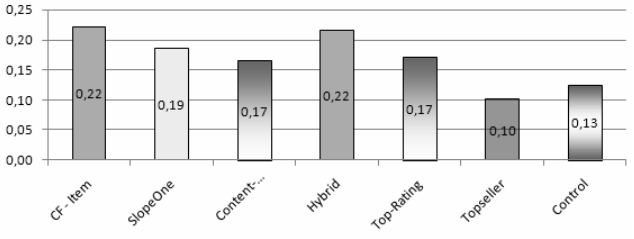


Figure 9: Item visits per post-sales list views

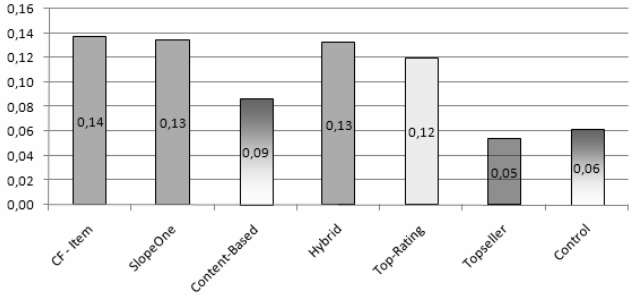


Figure 10: Item purchases to post-sales list visits

(Figure 7). When it comes to actual purchases (Figure 8), however, the manually-edited list did not help well to turn more visitors into buyers. Interestingly, the relative improvement caused by personalized recommendations with respect to this conversion rate is higher on the post-sales recommendation page than in the "My Recommendations" sections. Again, the CF algorithm worked best; in absolute numbers, the differences between the various techniques are significant, $p < 0.01$. With respect to the number of item visits and purchases per customer (Figures 9 and 10), it can again be observed that the different recommendation techniques not only stimulated visitors to view more items but actually also helped to increase sales. It can also be seen that displaying a list of top-selling items after a purchase leads to a particularly poor effect with respect to the overall number of downloads.

Another observation is that the items that are recommended by the SlopeOne technique and the TopRating method are also downloaded very often (see 10), presumably because the recommendation lists again contain many free demos. Figure 11 therefore shows the ratio of demo downloads to game purchases, which is quite similar to the one

from the "My Recommendations" section, i.e., recommending top-selling or newly released items does not stimulate additional interest in free evaluation versions (demo games). The trend toward interest in demo versions seems to be a bit more amplified than in the "My Recommendations" section, which indicates that after a purchase transaction, customers first have a look on another, but free, game.

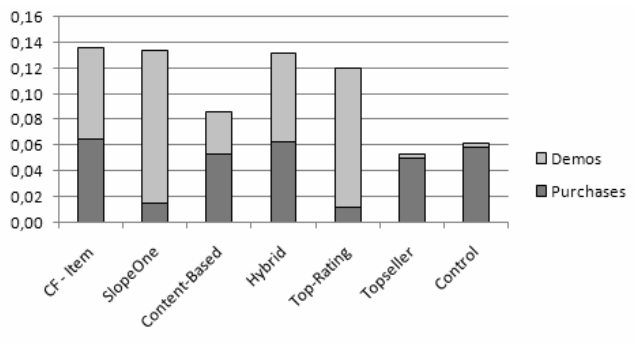


Figure 11: Game purchases and demo downloads on post-sales page

Finally, in this navigational context, the content-based method could raise some initial customer interest (Figure 9), perhaps because games are recommended that are quite similar to previously downloaded ones. However, while customers viewed some of the items, they had no strong tendency of purchasing them, probably because the games were – according to the general tendency of content-based methods – too similar to games they already know. The list of top selling items again contained mostly of non-free games, which explains the small fraction of demo games here; the same holds for the control group.

Measurement 3: Start-page recommendations

This measurement analyzes the effect of the personalized recommendations on the start page as shown in Figure 1. Remember that some elements in these lists are edited manually but were static during the experiment. Thus, we did not include item visits or purchases from these links (that could have been other banner advertisements as well) in the evaluation.

During the experiment, the personalized elements of the list, i.e., the last two text teasers and the first image teaser, were determined based on the top-3 list of the individual recommendation algorithms or based on the non-personalized lists of top-selling and top-rated items. Customers assigned to the control group received manually-determined recommendations which were ranked by release date.

For this experiment, we will only show the conversion rate figures for the different teaser elements on the start page.

Figure 12 shows the percentage of portal visitors that followed one of the personalized product links on the start page. On average, the image teaser was clicked on by around 6% of the users. Although the image only represents the third-ranked item of the recommendation algorithms and is also positioned after the text links, its conversion rate is significantly higher than for the text links. Since this also holds for the non-personalized methods, the attractiveness of the third link can be attributed to its visual representation. Interestingly, however, the image teaser leads to a good conversion rate with respect to actual sales (Figure 13). With respect to these conversion rates, both the CF method and the content-based method lead to a significant increase of item detail clicks and purchases. It can be also observed that the conversion rates of the first text teaser can even be better than the image teaser, when the text links are personalized. Thus, personalization can partially even outweigh the disadvantages of the unflashy representation.

Another particularity of this measurement on the start page is that the manually-selected items used for the control group lead to comparably good conversion rates, especially with respect to item visits. A possible explanation could be that customers have no special expectations with respect to the offers on the start page. The fact that the manually selected items are newly released ones might further contribute to the good acceptance.

Although recommending items based on their average customer rating (as done by the SlopeOne and the TopRating technique) worked well in the first two experiments, this approach does not work particularly well on the start page, i.e., customers seem to prefer either new items or items that are somehow related to their previous buying history.

Finally, when it comes to the number of purchases induced by the recommendation lists, the personalized techniques clearly outperformed the manually defined lists, at least for the first two teaser elements, see Figure 14.

Note that we also compared the item click and sales numbers of the other four and statically defined image and text teasers with the personalized ones. It could be seen that although the personalized items are partially placed lower on the screen and are thus harder to select, the received significantly more clicks and lead to more sales than the non-personalized links.

Measurement 4: Overall effect on demo downloads

In Measurement 1 and 2 we have seen that SlopeOne and the non-personalized technique based on item ratings lead to significantly more views and downloads of demo games. In this measurement, the goal was to analyze whether this trend also exists when the entire platform is considered, including, e.g., all other personalized and non-personalized navigation possibilities.

Note that no explicit category in the navigation tree for "free demos" exists. Games for which free evaluation versions exist, can however appear in all other personalized and non-personalized item listings in the portal. In addition, customers are pointed to demos in two additional ways: a) through direct-access links that are sent to them in sales promotions b) through pointers to other demo games that are displayed after a demo has been downloaded.

The distribution of views and downloads of demo games during the four-week evaluation period for the different recommendation groups is shown in Figure 15. Overall, about

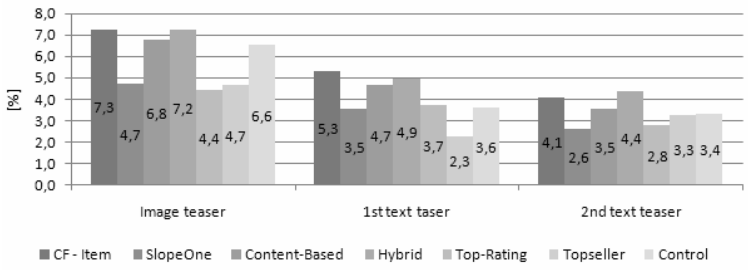


Figure 12: Conversion rate: Item views to start page visits

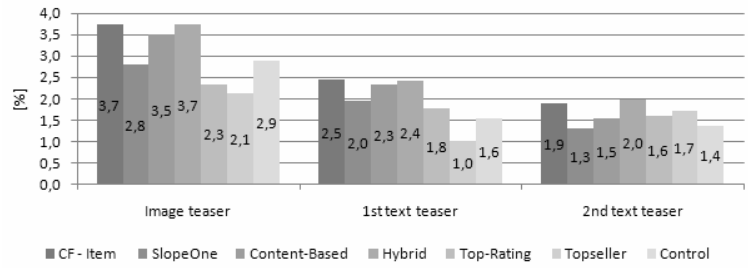


Figure 13: Conversion rate: Purchases from start page visits

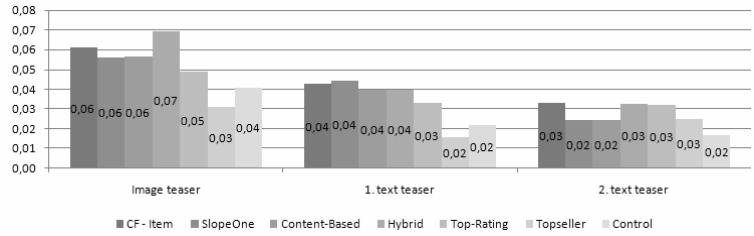


Figure 14: Purchases per start page visits

38.000 downloads have been observed for the selected subsets of customers. When considering the actual downloads, we see that the ranking of the algorithms remains the same; the differences are even amplified.

As already quickly mentioned in previous sections, this result can be explained by different facts that are related to the particular application setting and the nature of SlopeOne and the top-rating algorithm, which both tend to rank demo games highly in the different categories described above for the following reasons. First, as demo games can be downloaded at no cost and user ratings are only possible on the platform after a download, more explicit ratings are available for these games. Next, explicit ratings tend to be above-average also in this domain. Note that a similar phenomenon can also be observed in other datasets such as the MovieLens rating database. Finally, as customers receive a non-personalized pointer to another demos after downloading a free game, a reinforcement of the effect occurs.

An in-depth analysis, whether the downloads that were stimulated by the different algorithms lead to significantly *different* demo-download/purchase conversion rates is beyond the scope of the current study. What could, however, be observed in a first analysis is that the demo/purchase conversion rate is significantly higher when the demo was promoted by a recommendation list (as opposed to a banner advertisement).

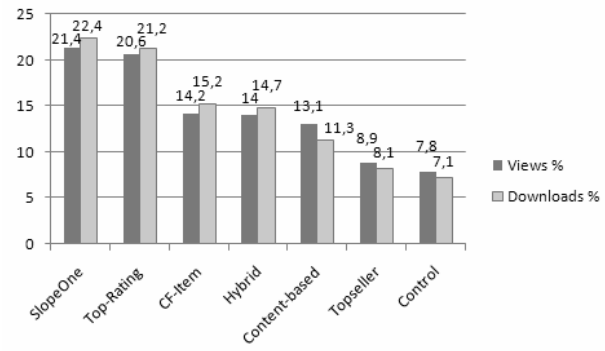


Figure 15: Distribution of demo game item views and downloads

Measurement 5: Overall effects

In this final measurement reported in this paper, the overall effect of the personalized recommendations (as an add-on to the other navigational options) situations was evaluated. Again, the interesting figures are related to item view and sales conversion rates (H1 and H2) as well as to the question whether more items are viewed and sold by individual customers (H3 and H4).

With respect to the conversion rates (hypotheses H1 and H2), no significant differences between the personalized and non-personalized variant could be observed, when the platform is considered as a whole. On average, about 80% of all observed customers have viewed at least one item and around 57% have bought at least one game, independent of the recommendation algorithm group they were assigned to. These figures are nearly identical for all seven test groups. For the item view conversion rate, for instance, the numbers only range from 79.6% to 80.3%. Thus, although slight improvements could be observed in individual (personalized) situations as described above, the influence on the overall conversion rate is too small and thus, the percentage of portal visitors that view or purchase items could not significantly be increased *by the additional use* of personalized recommendation lists.

There could be different reasons for this non-effect. First, remember that beside the described personalized lists, there are various other ways in which customers can access the item catalogs. Many customers for instance use the built-in search functionality of the portal; the ranking of search results is not personalized. The list of *new items* (see Figure 1) is also one of the most popular ways of browsing the catalog and used by significantly more people than, for instance, the new "My Recommendations" section. Our analysis shows that personalizing this particular list does not improve the conversion rates as customers always prefer to see the latest releases on top of such a list. Second, remember that only customers have been considered in the evaluation, for which a minimum number of rating already existed, i.e. users who are in generally interested in games. An evaluation of whether more *new users* can be tempted to purchase items was not in the focus of the evaluation.

With respect to the hypotheses H3 and H4 (increased number of item views and sales per customer), the following observations can be made. Regarding the average number of item views per customer (H3), we see that all personalized algorithms outperform the non-personalized topseller list and the control group. Similar to the effect of Measurement 4, SlopeOne and the simple ranking based on average customer rating raised the most attention. Thus, H3 could only partially be validated at the global scale as also the non-personalized top-rating technique was successful.

The observations made with respect to the number of purchased/downloaded items per customer (H4) are shown in Figure 16.

The figure shows that the additional attention raised by SlopeOne and the "Top Rating" algorithm also leads to an measurably increased number of items purchased and downloaded per customer. Figure 17 shows the number of downloaded items (including the demos) for the different algorithms. If we, finally, look at the actual sales numbers for non-free games only (Figure 18), we can see that although the Top-Rating list raised attention for free demos, it did not lead to increased sales for non-free items. Overall, all personalized techniques were more successful than the non-personalized one. On the global scale, the difference was however – a bit surprisingly – only significant for the content-based method, which indicates that customers tend to spend money on items that are similar to those they liked in the past. In fact, a closer look on the performance of the algorithms in specific sub-categories shows that the content-based method often slightly outperforms the other methods with respect to non-free games. While the differences were not significant in the individual situations – which is why we did not include these figures here – these slightly higher sales numbers sum up to a significant difference on the global scale. Examples of categories in which the content-based method worked slightly better with respect to non-free games are the "new games", "half-price", or "erotic games" section of the download portal.

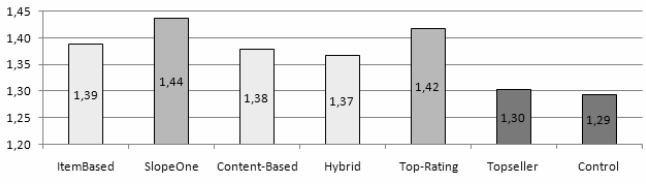


Figure 16: Average number of purchases including free downloads per customer on entire platform

Overall, the increase in actual sales that are directly stimulated by the recommender system is between 3.2% when compared to the Top-Rating technique and around 3.6% when no personalized recommendation is available.

In general, these last observations suggest that in situations where the user has no strong expectations on a certain genre (such as the "MyRecommendations" section), collaborative methods – which also recommend items of categories that the user has not seen before – work particularly well. In many other situations, however, users tend to prefer recommendations of game sub-categories that they already know. One exception is the post-sales situation, where users are, non-surprisingly, not interested in purchasing games which are very similar to the one he or she has bought a moment ago.

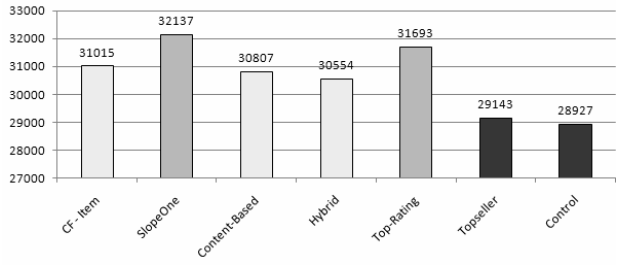


Figure 17: Total number of purchases and downloads

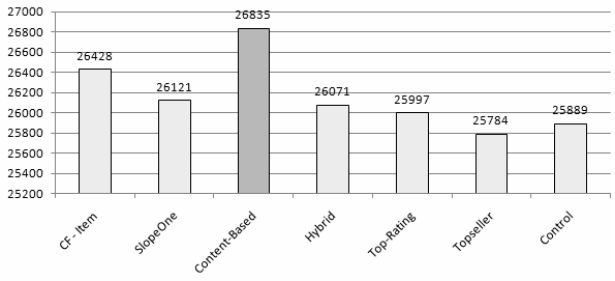


Figure 18: Total number of purchases (without demos)

Summary

In this work, the effects of personalized item recommendation in various navigational contexts on a Mobile Internet game portal were analyzed. Different standard recommendation techniques have been implemented on the portal and deployed in parallel in a real-world setting for a period of four weeks. In addition, non-personalized techniques based on top-selling or top-rated items have been used for comparison purposes.

The findings can be summarized as follows.

Ratings in the Mobile Internet

The amount of explicit item ratings was very low on the considered Mobile Internet portal and only about 2% of the users issued explicit ratings. While we are not aware of any studies that compare the willingness of customers to rate items in different settings, we suspect that the relatively high effort for submitting an item vote using a mobile device compared to a web browser discourages users to participate in this community process. When using only explicit ratings (in combination with a minimum number of neighbors as to decrease the Mean Absolute Error MAE), the *coverage* and applicability of individual algorithms quickly degrades. An analysis for the item-to-item algorithm for instance showed that the *item coverage* degrades to less than 50% when the minimum number of neighbors is set to the value of 20. In our experiments we therefore also took two types of implicit ratings into account: item views and item purchases, as the offline analysis showed that this combination leads to a good coverage (between 90% and 95%) for the item-based recommender. Note however that the MAE was also not very helpful to predict the accuracy of different algorithms in advance when including these implicit ratings. An offline evaluation showed that due to the high number of similar ratings very similar and low (below 0.2 points on the five-point scale) MAE values were achieved. An analysis of whether or not using different values for implicit ratings can help to further improve the recommendation accuracy, was not part of the current study.

Recommending in navigational context

In this study, the effects of personalized recommendations have been measured in different navigational situations such as the start page of the portal or the post-sales situation. In addition, we differentiated between the interest that was raised by the recommendations and the actual effect on the buying behavior of the customers.

With respect to the navigational context, customers seem to react slightly differently to recommendations, probably because of different expectations. In the dedicated "My Recommendations" section of the portal, classical collaborative filtering and the hybrid technique are particularly good at raising customer interest as customers view many of the recommended items. While customers are also easily stimulated to download free games by the comparably simple SlopeOne and TopRating method, these techniques do not lead to a significant increase in non-free games. A similar effect can be observed in the post-sales situation; the trend toward free demo downloads is even amplified in this situation. Thus, the item-based, content-based and hybrid technique which lead to a good number of purchases but also raise additional interest in demos, seems to be a good choice here.

On the portal entry page, the recommendation of top-rated items (or topsellers) has a particularly poor effect and the personalized methods lead to significantly better results. A listing of newly released items on the start page works however also quite well.

In certain navigational situations, we observed that personalization worsens the conversion rates and sales numbers. In the section on new items, which contains games of the last three weeks, the strict chronological order with the newest items on top works best. Most probably, the visitors of the "New" category enter this section regularly and only check the first few lines for new arrivals.

Finally, when measuring the number of game downloads including the demos on the entire platform, it shows that naive approaches such as TopRating and the comparably simple SlopeOne technique work sufficiently well to raise the users' interest in individual games. The important result, however, is that with respect to actual sales, the content-based and the item-based methods were clearly better than all others. Overall, it could be demonstrated that recommender systems are capable to stimulate a *measurable increase in overall sales* by over 3 percent on the entire platform.

Acknowledgement

The work was supported by Twistbox Games Ltd. & Co. KG, Germany. The described portal is based on Twistbox's Nitro-CDP™ content-download platform.

References

Billsus, D., and Pazzani, M. J. 2007. Adaptive news access. In Brusilovsky, P.; Kobsa, A.; and Nejdl, W., eds., *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, 550–570. Springer.

Breese, J. S.; Heckerman, D.; and Kadie, C. M. 1998. Empirical analysis of predictive algorithms for collaborative filtering. In Cooper, G. F., and Moral, S., eds., *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, 43–52.

Burke, R. 2002. Hybrid recommender systems: Survey and

experiments. *User Modeling and User-Adapted Interaction* 12(4):331–370.

Cho, Y. H.; Kim, C. Y.; and Kim, D.-H. 2004. Personalized image recommendation in the Mobile Internet. In *Procedings of 8th Pacific Rim International Conference on Artificial Intelligence*, volume 3157 of *Lecture Notes in Computer Science*, 963–964. Auckland, New Zealand: Springer.

Dias, M. B.; Locher, D.; Li, M.; El-Deredy, W.; and Lisboa, P. J. 2008. The value of personalised recommender systems to e-business: a case study. In *RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems*, 291–294.

Fleder, D. M., and Hosanagar, K. 2007. Recommender systems and their impact on sales diversity. In *EC '07: Proceedings of the 8th ACM conference on Electronic commerce*, 192–199.

Herlocker, J. L.; Konstan, J. A.; Terveen, L. G.; and Riedl, J. T. 2004. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.* 22(1):5–53.

Lemire, D., and Maclachlan, A. 2005. Slope one predictors for online rating-based collaborative filtering. In *Proceedings of SIAM Data Mining (SDM'05)*, 471–480.

Li, Q.; Wang, C.; and Geng, G. 2008. Improving personalized services in mobile commerce by a novel multicriteria rating approach. In *WWW'08: Proceeding of the 17th international conference on World Wide Web*, 1235–1236.

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

McNee, S. M.; Riedl, J.; and Konstan, J. 2006. Accurate is not always good: How accuracy metrics have hurt recommender systems. In *Extended Abstracts of the 2006 ACM Conference on Human Factors in Computing Systems (CHI 2006)*.

Miller, B. N.; Albert, I.; Lam, S. K.; Konstan, J. A.; and Riedl, J. 2003. Movielens unplugged: experiences with an occasionally connected recommender system. In *IUI '03: Proceedings of the 8th international conference on Intelligent user interfaces*, 263–266.

Nguyen, Q. N., and Ricci, F. 2008. Long-term and session-specific user preferences in a mobile recommender system. In *IUI '08: Proceedings of the 13th international conference on Intelligent user interfaces*, 381–384.

Pazzani, M. J. 2002. Commercial applications of machine learning for personalized wireless portals. In *Proceedings of the 7th Pacific Rim International Conference on Artificial Intelligence*, 1–5.

Ricci, F., and Nguyen, Q. N. 2007. Acquiring and revising preferences in a critique-based mobile recommender system. *IEEE Intelligent Systems* 22(3):22–29.

Sarwar, B.; Karypis, G.; Konstan, J.; and Reidl, J. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, 285–295.

Senecal, S., and Nantel, J. 2004. The influence of online product recommendations on consumers' online choices. *Journal of Retailing* 80(2):159 – 169.

Shani, G.; Brafman, R. I.; and Heckerman, D. 2002. An MDP-based recommender system. *Journal of Machine Learning Research* 6:453–460.

Smyth, B., and Cotter, P. 2002. Personalized adaptive navigation for mobile portals. In *Proceedings of the 15th European Conference on Artificial Intelligence*, 608–612.

Smyth, B.; Cotter, P.; and Oman, S. 2007. Enabling intelligent content discovery on the mobile internet. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, 1744–1751.

van der Heijden, H.; Kotsis, G.; and Kronsteiner, R. 2005. Mobile recommendation systems for decision making "on the go". In *ICMB '05: Proceedings of the International Conference on Mobile Business*, 137–143.

Zanker, M.; Bricman, M.; Gordea, S.; Jannach, D.; and Jessenitschnig, M. 2006. Persuasive online-selling in quality and taste domains. In *Proceedings of 7th Intl. Conference E-Commerce and Web Technologies*, 51–60.

Zanker, M.; Jessenitschnig, M.; Jannach, D.; and Gordea, S. 2007. Comparing recommendation strategies in a commercial context. *IEEE Intelligent Systems* 22(3):69–73.

Zanker, M.; Fuchs, M.; Höpken, W.; Tuta, M.; and Müller, N. 2008. Evaluating recommender systems in tourism - a case study from Austria. In *Proceedings Information and Communication Technologies in Tourism, ENTER 2008*, 24–34.

Adapting K-Nearest Neighbor for Tag Recommendation in Folksonomies

Jonathan Gemmell, Thomas Schimoler, Maryam Ramezani, Bamshad Mobasher

Center for Web Intelligence
School of Computing, DePaul University
Chicago, Illinois, USA

{jgemmell, tschimoler, mramezani, mobasher}@cdm.depaul.edu

Abstract

Folksonomies, otherwise known as Collaborative Tagging Systems, enable Internet users to share, annotate and search for online resources with user selected labels called tags. Tag recommendation, the suggestion of an ordered set of tags during the annotation process, reduces the user effort from a keyboard entry to a mouse click. By simplifying the annotation process tagging is promoted, noise in the data is reduced through the elimination of discrepancies that result in redundant tags, and ambiguous tags may be avoided. Tag recommenders can suggest tags that maximize utility, offer tags the user may not have previously considered or steer users toward adopting a core vocabulary. In sum, tag recommendation promotes a denser dataset that is useful in its own right or can be exploited by a myriad of data mining techniques for additional functionality.

While there exists a long history of recommendation algorithms, the data structure of a Folksonomy is distinct from those found in traditional recommendation problems. We first explore two data reduction techniques, p-core processing and Hebbian deflation, then demonstrate how to adapt K-Nearest Neighbor for use with Folksonomies by incorporating user, resource and tag information into the algorithm. We further investigate multiple techniques for user modeling required to compute the similarity among users. Additionally we demonstrate that tag boosting, the promoting of tags previously applied by a user to a resource, improves the coverage and accuracy of K-Nearest Neighbor.

These techniques are evaluated through extensive experimentation using data collected from two real Collaborative Tagging Web sites. Finally the modified K-Nearest Neighbor algorithm is compared with alternative techniques based on popularity and link analysis. We find that K-Nearest Neighbor modified for use with Folksonomies generates excellent recommendations, scales well with large datasets, and is applicable to both narrow and broadly focused Folksonomies.

Introduction

Folksonomies, also known as Collaborative Tagging Systems, have emerged as a powerful trend allowing Internet users to share, annotate and explore online resources through personalized labels. Several Collaborative Tagging Systems have recently gained popularity attracting millions of users. Delicious[1] supports users as they bookmark URLs. Flickr[2] allows users to upload, share and manage online photographs. Citeulike[3] enables researchers to manage and discover scholarly references. Still other Collaborative Tagging Systems specialize in music, blogs and business documents.

At the core of Collaborative Tagging Systems is the post: a user describes a resource with a set of tags. These tags may be descriptive ("Folksonomies"), subjective ("awesome"), organizational ("toread") or completely idiosyncratic ("jfgwh"). Taken in isolation an individual annotation allows a user to organize web resources for later use: resources can be easily sorted, aggregated and retrieved. Resources may be annotated with multiple tags allowing a resource to be identified with several topic areas rather than pigeonholed in a single directory. Moreover users need not conform to a predefined vocabulary or rigid hierarchy but may annotate a resource with any tag they wish thereby reducing user effort and limiting the entry cost.

However the utility of tagging extends beyond their immediate use. Taken as a whole, the aggregation of many annotations results in a complex network of interrelated users, resources and tags often referred to as a Folksonomy (Mathes 2004). The opportunity to explore the Folksonomy unburdened by a preconceived navigational or conceptual hierarchy is crucial to the utility and popularity of Folksonomies. Users are able to share or discover resources through the collaborative network and connect to other users with related interests. Collaborative Tagging Systems can identify groups of like-minded users, catering not only to mainstream but also to non-conventional users who are often under-served by traditional Web tools. Furthermore, users may enjoy the social aspects of collaborative tagging, attracted by a sense of community not offered by either ontologies or search engines.

A distinct advantage of Folksonomies is the richness of the user profiles. If a user is interested enough in a resource to annotate it, the tag describes the user as much as it describes the resource. As users annotate resources, the system is able to track their interests. These profiles are a powerful tool for data mining algorithms.

[1] delicious.com
[2] www.flickr.com
[3] www.citeulike.org

Even though tags offer many benefits both in the short and long term, they also present unique challenges for recommendation algorithms. Most Collaborative Tagging Applications permit unsupervised tagging; users are free to use any tag they wish to describe a resource. This is often done to reduce the entry cost of using the application and make collaborative tagging more user-friendly. Unsupervised tagging can result in tag redundancy in which several tags have the same meaning or tag ambiguity in which a single tag has many different meanings. Such inconsistencies can confound users as they attempt to utilize the Folksonomy.

Tag recommendation provides a means to overcome these problems. It reduces the user effort to a mouse click rather than a keyboard entry. By reducing the effort users are encouraged to tag more frequently, apply more tags to an individual resource, reuse common tags, and perhaps use tags the user had not previously considered. Moreover, user error is reduced by eliminating redundant tags caused by capitalization inconsistencies, punctuation errors, misspellings and other discrepancies. The tag recommender can further promote a core tag vocabulary steering the user toward adopting certain tags while not imposing any strict rules. The tag recommender may even avoid ambiguous tags in favor of tags that offer greater information value. The final result is a cleaner, denser dataset that is useful in its own right or for further data mining techniques.

However the data generated through Collaborative Tagging differs from that common in recommendation algorithms. The introduction of tags manifests a third dimension which must be integrated into recommenders that traditionally incorporate only two dimensions. In this paper we demonstrate how K-Nearest Neighbor may be adapted to recommend tags in Folksonomies. We describe how both user and resource information can be directly applied in the algorithm improving both coverage and accuracy while reducing computational costs. In addition several user models are explored including vectors over the set of tags, vectors over the set of resources, combinations of these two and features derived through Hebbian deflation.

The rest of this paper is organized as follows. We begin by presenting some related work involving the use of recommendations in Folksonomies. We explore the data structure of folksonomies and discuss two feature reduction techniques: p-core processing and Hebbian deflation. We then outline the basic approach used for recommending tags in Folksonomies and propose modifications to K-Nearest Neighbor. After a discussion of the datasets and a description of the experimental methodology, we evaluate the proposed modifications using data collected from two real world Folksonomies. Finally, we compare the modified algorithm with alternative strategies based on popularity and link analysis.

Related Work

As Collaborative Tagging Applications have gained in popularity researchers have started to explore and characterize the tagging phenomenon. In (Macgregor and McCulloch 2006) and (Golder and Huberman 2006) the authors studied the information dynamics of Delicious, one of the most popular Folksonomies. The authors discussed how tags have been used by individual users over time and how tags for an individual resource stabilize over time. They also discussed two semantic difficulties: tag redundancy, when multiple tags have the same meaning, and tag ambiguity, when a single tag has multiple meanings. (Macgregor and McCulloch 2006) provide an overview of the phenomenon and explore reasons why both Folksonomies and Ontologies will have a place in the future of information access.

There have been several recent research investigations into recommendation within Folksonomies. Unlike traditional recommender systems which have a two-dimensional relation between users and items, tagging systems have a three dimensional relation between users, tags and resources. Recommender systems can be used to recommend each of the dimensions based on one or two of the other dimensions. (Tso-Sutter, Marinho, and Schmidt-Thieme 2008) applies user-based and item-based collaborative filtering to recommend resources in a tagging system and uses tags as an extension to the user-item matrices. (Nakamoto et al. 2008a) and (Nakamoto et al. 2008b) use tags as context information to recommend resources.

Other researchers have studied tag recommendation in folksonomies. (Jaschke et al. 2007) compares user-based collaborative filtering and a graph-based recommender based on the Pagerank algorithm to recommend personalized tags. (Heymann, Ramage, and Garcia-Molina 2008) use association rules to recommend tags and introduce an entropy-based metric to find how predictable a tag is. (Lipczak 2008) uses the title of a resource, the posts of a resource and the user's vocabulary to recommend tags. The results show that tags retrieved from the user's vocabulary outperform recommendations driven by resource information. However the experiment was performed on data from Bibsonomy, a Folksonomy focused on scientific publications, and thus might not be applicable to multi-domain data that cover.

(Xu et al. 2006) presents general criteria for a good tagging system including high coverage of multiple facets, high popularity and least-effort. They categorize tags to content-based tags, context-based tags, attribute tags, subjective tags, and organizational tags and use a probabilistic method to recommend tags. (Basile et al. 2007) proposes a classification algorithm for tag recommendation. (Sigurbjörnsson and van Zwol 2008) uses a co-occurrence-based technique to recommend tags for photos in Flickr. The assumption is that the user has already assigned a set of tags to a photo and the recommender uses those tags to recommend more tags. (Adrian, Sauermann, and Roth-Berghofer 2007) suggests a semantic tag recommendation system in the context of a semantic desktop. (Song et al. 2008) uses clustering to make real-time tag recommendation.

Data Structures of Folksonomies

In this section we define the three-dimensional data structure of a Folksonomy and describe how to construct two-dimensional projections. We further explore two data reduction techniques for use with Folksonomies. The first is a data selection technique, P-core processing, that reduces the size

of the data by removing users, resources and tags that occur less than a predefined number of times. The second is a data extraction technique, Hebbian Deflation, which produces a new set of features from a two-dimensional projection of the Folksonomy.

Data Models

The data structure of a Folksonomy differs from that common to most traditional recommendation algorithms. A Folksonomy can be described as a four-tuple D:

$$D = \langle U, R, T, A \rangle , \tag{1}$$

where, U is a set of users; R is a set of resources; T is a set of tags; and A is a set of annotations, represented as user-tag-resource triples:

$$A \subseteq \{\langle u, r, t \rangle : u \in U, r \in R, t \in T\} \tag{2}$$

A Folksonomy can, therefore, be viewed as a tripartite hyper-graph (Mika 2007) with users, tags, and resources represented as nodes and the annotations represented as hyper-edges connecting a user, a tag and a resource.

The tripartite nature of Folksonomies make it ill suited for many traditional data mining techniques. User based K-Nearest Neighbor for example requires a means to measure the similarity between users. The introduction of a third dimension confounds this task.

Aggregate projections of the data can be constructed, reducing the dimensionality by sacrificing information. (Schmitz et al. 2006) The relation between resources and tags can be formulated as a two-dimensional projection, RT, such that each entry, $RT(r, t)$, is the weight associated with the resource, r, and the tag, t. This weight may be binary, merely showing that one or more users have applied that tag to the resource, or it may be finer grained using the number of users that have applied that tag to the resource:

$$RT_{tf}(r, t) = |\{a = \langle u, r, t \rangle \in A : u \in U\}| \tag{3}$$

Such a measure is equivalent to *term frequency* or *tf* common in Information Retrieval. Similarly, *term frequency * inverse document frequency* or *tf*idf* (Salton and Buckley 1988) can be adapted for use with two-dimensional projections:

$$RT_{tf*idf}(r, t) = RT_{tf}(r, t) * \log(|R|/n_r) \tag{4}$$

The *tf*idf* multiplies the tag frequency by the relative distinctiveness of the tag. The distinctiveness is measured by the log of the total number of resources, $|R|$, divided by the number of resources to which that tag was applied, n_r. Similar two-dimensional projections can be constructed for *UT* in which the weights correspond to users and tags, and *UR* in which the weights correspond to users and resources.

While a portion of the information is lost through this process the result is a two-dimensional matrix which can be readily applied to existing data-mining algorithms such as K-Nearest Neighbor.

P-Core Processing

Through P-Core Processing users, resources and tags are removed from the dataset in order to produce a residual dataset that guarantees each user, resource and tag occur in at least p posts (Batagelj and Zaveršnik 2002) (Jaschke et al. 2007). Here we define a post to include a user, a resource, and every tag the user has applied to the resource.

The algorithm iterates through the posts, counting the occurrence of users, resources and tags. If the occurrence of these items does not meet the requisite value, p, all occurrences of the item and the posts in which it appears are removed from the dataset. Since removing a post based on the occurrence of one item reduces the count for the other items in the post, several passes through the dataset may be required. The result is a smaller denser dataset.

Several reasons exist to use P-Core Processing. By removing infrequent users, resources and tags noise in the data is reduced; uncommon items whether they be tags used by only a few users, unpopular resources, or unproductive users are eliminated from consideration. Because of their scarcity, these are the very items likely to confound recommenders. Moreover, by eliminating infrequent items, the size of the data may be dramatically reduced allowing the application of the data mining techniques that would otherwise be computationally impractical. In short, P-Core Processing offers a means reduce noise and focus on the dense regions of the data.

Hebbian Deflation

Whereas P-Core Processing offers a means to reduce the size of a dataset through feature selection, feature extraction techniques generate entirely new features. Many feature extraction techniques exist such as Principle Component Analysis and Singular Value Decomposition. However, despite the utility these techniques provide, their computational cost and memory requirements make them impractical for use on extremely large datasets common in Folksonomies. Hebbian Deflation (Oja and Karhunen 1985) offers a means to approximate these feature extraction techniques with reduced computational and memory needs.

Given a two-dimensional projection of the Folksonomy and a preselected number of features, *F*, Hebbian Deflation produces two smaller matrices that approximate the original projection. Consider for example the two-dimensional projection *RT*; Hebbian Deflation will produce two new matrixes, *RX* and *TX*, such that:

$$RT(r, t) \approx \sum_{i=0}^{F} RX_{ri} * TX_{ti} \tag{5}$$

Since *F* is often far smaller than either the number of resources or the number of tags the resultant matrixes require many orders of magnitude less space than the original projection. Analogous features can be extracted from *UR* and *UT*.

The algorithm requires many inputs: F, the number of features to be extracted; $epochs$ the number of epochs to train each feature; and lr the learning rate at which the features are adjusted.

Features are extracted one at a time. To begin, the features are set to random weights. Then, for each resource-tag pair in RT the actual weight is compared with the current approximation. The features are adjusted based upon the degree of the error, the learning rate and the corresponding feature in the opposing matrix. These adjustments are repeated for the predetermined number of epochs before the feature is finally adopted and the next feature is trained.

Input: RT, a projection of a Folksonomy; F, the number of features to derive; $epochs$, the number of epochs to train each feature; lr, the learning rate
Output: RX and TX, the Hebbian features
for $f = 1 \rightarrow features$ **do**
  for $epoch = 1 \rightarrow epochs$ **do**
    foreach $(r, t) \in RT$ **do**
      $aprox = \sum_{i=1}^{f} RX_{ri} * TX_{ti}$
      $error = RT(r, t) - approx$
      $RX(r, f)$ += $lr * error * TX(t, f)$
      $TX(t, f)$ += $lr * error * RX(r, f)$
    end
  end
end
return RX and TX;

Algorithm 1: Hebbian Deflation

Like many learning algorithms, Hebbian Deflation may result in overfitting. Overfitting occurs when the features over specialize in the training data at the expense of generalization. In order to combat this, a percentage of the training data may be held out. At each epoch, the Root Mean Square Error (RMSE), is calculated both for the ability of the features to approximate the training data and for their ability to approximate the holdout set. If it is observed that the RMSE is rising for the holdout data even as it is dropping for the training data, overfitting can be assumed. Then the training of the current feature may be halted and the training of the next feature can begin.

Feature extraction through Hebbian Deflation offers many benefits. It may offer a means to represent the data in a smaller space, but the features themselves may offer insights into the data. Domain experts can analyze features for their characteristics. Users may navigate over the reduced feature space rather than the larger Folksonomy. Users, resources and tags may be modeled as a vector over the set of features allowing reduced computation. Finally, Hebbian Deflation may reveal similarities among items in a Folksonomy that remained hidden when the data was expressed as a projection using *tf* or *tf*idf*.

Tag Recommendation in Folksonomies

Recommendation algorithms serve a vital role in Web applications allowing users to focus on a few relevant items rather than being overwhelmed by a large unordered set of mostly inappropriate options. Tag recommendation in Folksonomies reduces the user effort to a mouse click rather than a keyboard entry. This reduction in effort encourages tagging more resources, promotes the application of multiple tags to a resource, and may present the user with useful tags the user had not considered. Moreover by eliminating the keyboard entry tag recommendation reduces capitalization inconsistencies, punctuation errors, misspellings, and other discrepancies that add noise to the data.

The recommendation of high quality tags benefits the user beyond the annotation process itself. Resources are often tagged for future reference; by providing the most relevant tags retrieval of the resources are made easier. Moreover, when resources are tagged in order to characterize content, the recommendation of clear descriptive tags can improve the online experience for other users that are navigating the site. This is particularly relevant for resources that are not easily evaluated by computers such as photos, videos, and music.

Tag recommendation may also be used to assert control on tag usage without encumbering the user with a strict vocabulary. Ambiguous tags such as can be avoided in preference of less ambiguous tags Moreover, the recommender can offer tags with a higher level of detail. The overuse of redundant tags can also be thwarted by consistently recommending highly used tags while eschewing their less used counterparts. The result is cleaner denser dataset that is useful in own right for navigation, or for further data mining techniques.

Basic Recommendation

In traditional recommendation algorithms the input is often a user, u, and the output is a set of items, I. The user experience is improved if this set of items is relevant to the user's needs.

Tag recommendation in Folksonomies however differs in that the input is both a user, u, and a resource, r. The output remains a set of items, in this case a recommended set of tags, T_r. One of the difficulties presented by tag recommendation is the means to incorporate both user and resource information into the recommendation algorithm.

Perhaps the simplest recommendation strategy is merely to recommend the most commonly used tags in the Folksonomy. However such a strategy ignores both user and resource information.

Alternatively given a user-resource pair a recommender may ignore the user and recommend the most popular tags for that particular resource. This strategy is strictly resource dependent and ignores the tagging habits of the user. In a similar fashion a recommender may ignore the resource and recommend the most popular tags for that particular user. While such an algorithm would include tags frequently applied by the user, it ignores the resource information and may recommend tags irrelevant to the current resource.

An algorithm for tag recommendation in Folksonomies therefore requires a means to include both user and resource information in the process so that the recommendation set includes tags that are relevant to the resource and also represent the user's tagging practice.

K-Nearest Neighbor

User Based K-Nearest Neighbor is a commonly used recommendation algorithm in Information Retrieval that can be

modified to include both user and resource information. Traditionally it finds a set of users similar to a query user. From these neighbors a set of recommended items it constructed.

We can modify this approach by ignoring users that have not tagged the query resource. Once a neighborhood of similar users has been discovered, the algorithm considers only on those tags that have been applied to the query resource and calculates a weight for each tag, w_t, the average similarity of the neighbors that have applied the tag to the query resource. Thus the algorithm is resource driven through both the selection of neighbors and the selection of tags. Still it remains user driven in that neighbors are determined through a user model.

Input: u_q, a query user; r_q, a query resource; k, number of neighbors to consider; n, the number of tags to recommend
Output: T_r, a set of recommended tags
foreach $u \in U$ *that has annotated* r_q **do**
  s_u = similarity(u, u_q)
end
Let N be k nearest neighbors to u_q;
foreach $u \in N$ **do**
  foreach t *that* u *applied to* r_q **do**
    w_t += s_u/k
  end
end
Sort tags by w_t;
Let T_r be the top n tags;
Return T_r

Algorithm 2: K-Nearest Neighbor Modified for Folksonomies

K-Nearest Neighbor is considered a lazy algorithm; the bulk of its computation takes place after the query. Traditional approaches would require a comparison between the query user and every other user. However, since the adapted algorithm for K-Nearest Neighbor considers only those users that have annotated the query resource, the number of similarities to calculate is drastically reduced. The popularity of resources in Folksonomies follows the power law and the great majority of resources will benefit from this reduced reduction in computation, while a few will require additional computational effort. As a result the adapted K-Nearest Neighbor scales well with large datasets, a trait not shared by many other recommendation algorithms.

User Models

Applications vary in the way they model users. Possible methods include recency, authority, linkage or vector space models. In this work we focus on the vector space model (Salton, Wong, and Yang 1975) adapted from the Information Retrieval discipline to work with Folksonomies. Each user, u, can be modeled as a vector over the set of tags, where each weight, $w(t_i)$, in each dimension corresponds to the importance of a particular tag, t_i.

$$\vec{u_t} = \langle w(t_1), w(t_2)...w(t_{|T|})\rangle \tag{6}$$

In calculating the vector weights a variety of measures can be used: binary, *term frequency* or *term frequency*inverse document frequency*. In this work we focus on *term frequency*. Similarly a user can be modeled as a vector over the set of resources where each weight, $w(r_i)$, corresponds to the importance of a particular resource, r_i.

$$\vec{u_r} = \langle w(r_1), w(r_2)...w(r_{|R|})\rangle \tag{7}$$

Both of these models however ignore a portion of the user profile. A user model consisting merely of tags does not consider to which resources those tags have been applied. And a user model consisting only of resources does not include the tags applied to them.

The user model may be extended to include both tags and resources. A new vector can be obtained by concatenating the two previously mentioned vectors.

$$\vec{u_{t+r}} = \langle w(t_1)...w(t_{|T|}), w(r_1)...w(r_{|R|})\rangle \tag{8}$$

While this model does include both tags and resources, the model does not specify which tags were applied to which resources. However, the tags and resources may be tightly coupled in a vector over all tag-resource pairs where each weight, $w(tr_i)$, is one if the user has applied tag, t, to the resource, r, and zero otherwise.

$$\vec{u_{(tr)}} = \langle w(t_1r_1), w(t_1r_2)...w(t_{|T|}r_{|R|})\rangle \tag{9}$$

However these user models risk becoming exceedingly large and extremely sparse. Hebbian features may be used to combat this sparsity. For example, features extracted from either *UR* or *UT* may be used to construct the user model:

$$\vec{u_{Hebbian}} = \langle f_1, f_2...f_{|F|}\rangle \tag{10}$$

These extracted features can greatly reduce the computational costs of calculating similarities since the number of features is far smaller than the size of the original matrix. Moreover feature extraction may discover hidden relationships and identify similarities among users that the previously described models would not capture.

Several techniques exist to calculate the similarity between vectors such as Jaccard similarity or Cosine similarity (Van Rijsbergen 1979). In this work we focus on cosine similarity.

Boosting Tags

In most traditional recommendation approaches the recommendation set would not include an item the user has already used. However in Collaborative Tagging Applications users often reuse tags. Previously used tags are then an important clue for the recommendation algorithm.

We propose a boosting factor, b, that can be used to promote tags in the user profile. As an additional step to the modified K-Nearest Neighbor recommender, b is added to the weight of the tag if the user has previously applied that tag to another resource.

foreach $t \in T$ *that any* $u \in N$ *has applied to* r_q **do**
  if *(*u_q *has applied* t*)* **then**
    $w_t = w_t + b$
  end
end

Algorithm 3: Optional Step Including Boost K-Nearest Neighbor

FolkRank

In (Hotho et al. 2006) the authors proposed an adaptation of link analysis to the Folksonomy data structure. They have called this technique Folkrank since it computes a Pagerank vector from the tripartite graph induced by the Folksonomy. This graph is generated by regarding $U \cup R \cup T$ as the set of vertices. Edges are defined by the two-dimensional projections, *UT*, *UR* and *RT*.

If we regard the adjacency matrix of this graph, W, (normalized to be column-stochastic), a damping factor, d, and a preference vector, p, then we compute the Pagerank vector, w, in the usual manner:

$$w = dAw + (1 - d)p \tag{11}$$

However due to the symmetry inherent in the tripartite graph, this basic Pagerank can too easily focus on the most popular elements in the Folksonomy. The Folkrank vector is taken as a difference between two computations of Pagerank: one with a preference vector and one without the preference vector.

In order to generate tag recommendations Folkrank utilizes the preference vector to bias the algorithm towards the query user and resource(Jaschke et al. 2007). These elements are given a substantial weight in the preference vector where all other elements have uniformly small weights.

We have included this method as a benchmark as it has been shown to be an effective method of generating tag recommendations. However it has a distinct disadvantage in that it requires a complete computation of the Pagerank vector for each query. This makes the method problematic when working with data from large Folksonomies.

Experimental Evaluation

Here we describe the methods used to gather data for the experiments and provide details of our datasets. We then discuss modifications to N-Fold Cross Validation for Folksonomies and describe our experimental methodology. We briefly discus the common metrics recall and precision and then detail the results of our experiments.

Data Sets

We validate our approach through extensive evaluation of the proposed modifications using data from two real Collaborative Tagging Applications: Delicious and Citeulike.

Delicious is a popular Website in which users annotate URLs. On 10/19/2008, 198 of the most popular tags were taken from the user interface. For each of these tags the 2,000 most recent annotations including the contributors of the annotations were collected. This resulted in 99,864 distinct usernames.

For each user, the "Network" and "Fans" were explored recursively collecting additional usernames. A user's Network consists of the other users that the user has explicitly chosen to watch. Conversely a Fan is another user that has explicitly chosen to watch the user. This resulted in a total of 524,790 usernames.

From 10/20/2008 to 12/15/2008 the complete profiles of all 524,790 users were collected. Each user profile consisted of a collection of annotations including the resource, tags and date of the original bookmark. The top 100 most prolific users were visually inspected; twelve were removed from the data because their annotation count was many orders of magnitude larger than other users and were therefore suspected to be Web-bots.

Due to memory and time constraints, 10% of the user profiles was randomly selected. A P-core of 20 was derived such that each user, resource and tag appear in at least 20 posts where a post is defined as a user, resource and all tags that user applied to the resource.

The result was a dataset with 18,105 users, 42,646 resources and 13,053 tags. There are 2,309,426 annotations and 8,815,545 triples. The average number of tags in a post is 3.82.

Citeulike is a popular online tool used by researchers to manage and discover scholarly references. They make their dataset freely available to download[4]. On 2/17/2009 the most recent snapshot was downloaded with data extending back to 5/30/2007. The data contains anonymous user ids and posts for each user including resources, the date and time of the posting and the tags applied to the resource. The original dataset contains 41,689 users, 1,370,729 resource and 284,389 tags.

Because of its relatively small size and sparse data, the Citeulike data cannot support a P-core of 20. Instead a P-core of 5 was derived. This reduced the size of the data to 2,051 users, 5,376 resource and 3,343 tags. There are 42,277 annotations and 105,873 triples. The average number of tags in a post is 2.50.

Folksonomy	Delicious	Citeulike
Users	18,105	2,051
Resources	42,646	5,376
Tags	13,053	3,343
Posts	2,309,427	42,277
Annotations	8,815,545	105,873

Table 1: Datasets

An important distinction between the two datasets is their focus. Users in Delicious are able to tag any URL available on the Web. As such an individual's interests are often varied encompassing many topics. In Citeulike however researchers tag scholarly publications and their tagging is often focused in their area of expertise.

[4]http://www.citeulike.org/faq/data.adp

The data available for Delicious is also far more abundant. Using only a fraction of the data scraped from the Website, the Delicious dataset still has more than fifty times the annotations in the Citeulike dataset. Moreover, the Delicious is far denser supporting a P-core of 20 rather than a P-core of 5.

Experimental Methodologies

We implemented an extension of N-Fold Cross Validation for Folksonomies. Each user profile was divided among n folds, each fold containing approximately $1/n$ of each user's posts. A post includes the user, a resource and all tags the user applied to that resource. Models were built using $n-1$ folds of the data, while the posts in the remaining fold served as test cases.

Each test case consists of a user, u, a resource, r, and all the tags the user has applied to that resource. These tags, T_h, are analogous to the holdout set commonly used in Information Retrieval. The tag recommendation algorithms accept the user-resource pair and return an ordered set of recommended tags, T_r. From the holdout set and recommendation set utility metrics were calculated.

For each evaluation metric the average value was calculated across all test cases of an individual fold. The average was then calculated across all folds. Experiments completed on Delicious consisted of 10 folds, while experiments on Citeulike had 5 folds.

The exception to this methodology are the experiments completed for Folkrank. Due to the steep computational required for this approach only one post from each user was placed in the testing set. Experiments were then run on this single testing set as described in (Hotho et al. 2006).

Experimental Metrics

Recall is a common metric for evaluating the utility of recommendation algorithms. It measures the percentage of items in the holdout set that appear in the recommendation set. Recall is a measure of completeness and is defined as:

$$recall = |T_h \cap T_r|/|T_h| \tag{12}$$

Precision is another common metric for measuring the usefulness of recommendation algorithms. It measures the percentage of items in the recommendation set that appear in the holdout set. Precision measures the exactness of the recommendation algorithm and is defined as:

$$precision = |T_h \cap T_r|/|T_r| \tag{13}$$

Hebbian Features

A fundamental assumption of user models based upon Hebbian Deflation is that meaningful features can be extracted from the two-dimensional projections. In order to affirm this assumption we have taken the two-dimensional projection of the Delicious dataset, ***RT***, using the tag counts as the weights and performed Hebbian Deflation to generate RX and TX. A learning rate of .001 was chosen as this value allows the features to converge quickly and smoothly. Initially 100 features were built, but examination of the RMSE on a 2% overtraining-holdout set showed little improvement over 50 features. Additionally, 2000 epochs were selected to train each feature, but in all cases the algorithm halted the training when it detected overfitting and proceeded to the next feature.

Table 2 shows selected tags along with their six most similar neighbors. (Salton, Wong, and Yang 1975) has demonstrated similar results using co-occurrence, Folkrank and context metrics. Similarities are calculated by representing each tag as a vector of weights over the Hebbian features and calculating the cosine similarity between two tags.

Initial examination of the nearest tags lends credence to the assumption that Hebbian Deflation can be used to discover meaningful features. For example the nearest tags to "photo" are clearly redundant tags. However other techniques such as stemming and thesaurus tables could provide the same utility.

The example "toread" demonstrates that this method goes beyond what other techniques might provide. Where "photo" had been a descriptive tag, "toread" is an organizational tag. The ability of the Hebbian features to match it with things that will in fact be read and other organization tags illustrated the effectiveness of Hebbian Deflation.

Moreover in the example of "folksonomies" show how highly related words can be discovered through the Hebbian features. "Tags" are of course a crucial part of "folksonomies" that provide "classification" by providing "metadata."

Domain specific words like "mac" or "osx" are difficult to relate, but again Hebbian features are able to highlight their similarity. Perhaps the most intriguing example is the subjective tag "cool." Hebbian features are able to find two other subjective tags with high similarity, "interesting" and "fun".

Tags can also be modeled from features extracted from *UT*; Similarly users and resources may be modeled from features extracted from the projections. In all cases similar trends are observed.

Table 3 shows the same experiment completed with data from Citeulike. Again RT matrix was built from count data. The learning rate is set to .001. As before, 2000 epochs were selected, but RMSE testing on a holdout set halted the training of features when overfitting was detected. 100 features were extracted, but only the first 20 showed progress in reducing RMSE.

The related tags in Citeulike show similar trends to those found in Delicious. However because of its sparsity and relatively small size, it appears to be more difficult to extract features. Nevertheless visual inspection of the tags reaffirms the assumption that Hebbian Deflation and cosine similarity offer a method to discover meaningful features.

While this paper proposes using Hebbian features to model users, these features offer many other uses for Folksonomies. Hebbian features might provide a means for users to navigate the Folksonomy. After selecting a user, resource or tag the system could present additional items based upon the Hebbian features. The user may then select an item from that list and explore other users, resources or tags that are similar. By repeating this process the user could traverse

photo	shopping
0.789 photos	0.803 Shopping
0.737 photography	0.780 shop
0.652 pictures	0.560 buy
0.640 foto	0.530 google
0.637 Photo	0.519 store
0.627 fotos	0.469 handmade
toread	**folksonomy**
0.886 article	0.807 tagging
0.816 articles	0.786 tags
0.811 advice	0.691 tag
0.810 Bookmarks	0.635 classification
0.808 to_read	0.574 metadata
0.805 interesting	0.523 folksonomies
mac	**cool**
0.849 osx	0.711 interesting
0.840 Mac	0.673 fun
0.778 apple	0.621 imported
0.768 OSX	0.608 how-to
0.653 macosx	0.595 article
0.650 Apple	0.571 useful

Table 2: Selected Delicious tags and their nearest neighbors using cosine similarity and 50 Hebbian features extracted from the RT_c matrix

datamining	networks
0.981 computational	0.840 social_networks
0.935 conceptual	0.792 classification
0.933 data-mining	0.751 community
0.930 webservices	0.740 socialnetworks
0.925 tools	0.738 graph
0.915 data_mining	0.737 functional

Table 3: Selected Citeulike tags and their nearest neighbors using cosine similarity and 20 Hebbian features extracted from the RT_c matrix

the Folksonomy or focus in on a particular domain. Hebbian features might be particularly useful in this task since they are able to uncover relationships in the Folksonomy that other methods, such as co-occurrence, are unable to discover.

The individual features themselves might be interesting. A domain expert could analyze the features in an effort to understand how the Folksonomy is growing, what aspects are dominating, and which users are having the most impact.

Moreover this reduced yet rich feature space can be utilized in a variety of data mining tasks: recommendation, personalization, search, navigation, etc.

Experimental Results

Here we present our experimental results beginning with the tuning of variables. We discuss the impact of the boost variable in the quality of the K-Nearest Neighbor algorithm, then provide an in depth comparison of the recommendation techniques.

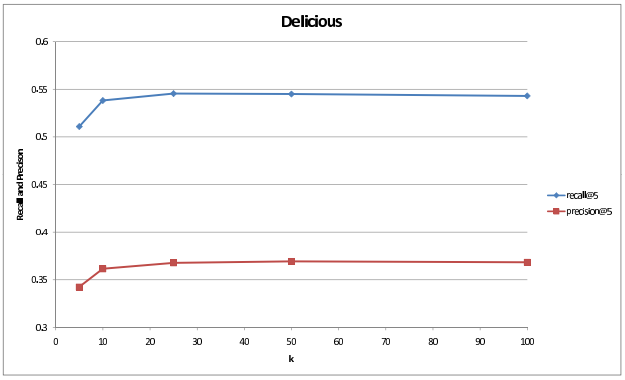


Figure 1: The effect of k in KNN on recall and precision for a recommendation set of 5 tags. Users are modeled as a vector over the tag space.

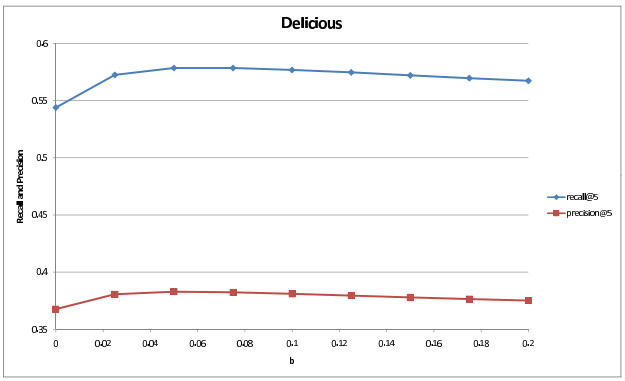


Figure 2: The effect of boosting previously used tags in KNN on recall and precision for a recommendation set of 5 tags. Users are modeled as a vector over the tag space.

The experiments with K-Nearest Neighbor require the tuning of two key variables: k, the number of neighbors, and b, the boosting factor.

Figure 1 shows the relation between k and the the evaluation metrics recall and precision for a recommendation set of size 5. The Delicious dataset was used for this experiment. Weights in the user vector are calculated as the frequency of tags or tf. This experiment does not include any boosting factor for previously used tags. As k increases so does recall and precision. However this improvement suffers from diminishing returns until a k of 100 offers little more benefit than a k of 50. This trend was observed for all K-Nearest Neighbor experiments. As such, all K-Nearest Neighbor experiments were completed using a k of 50. Similar results are observed for the Citeulike dataset.

Figure 2 demonstrates the effectiveness of the boosting modification for K-Nearest Neighbor. The modification gives extra weight to those tags the user has previously applied to a resource. This experiment is completed with a k of 50; b is adjusted in the range of 0 through 0.20 at 0.025 increments. Both recall and precision for a recommendation

set of 5 tags are sharply improved when b in increased to 0.05. Afterward, the effect of the boosting parameter slowly diminishes.

Table 4 provides a more detailed view of the effect boosting can have. For example without any boosting factor the precision for a recommendation set of size 3 is 43.5%. With the boosting factor, precision is increased to 45.9%, an immediate 3.4% gain. For all size recommendation sets precision is increased. The boosting factors enables K-Nearest Neighbor to become more precise.

Likewise recall increases across the board. For example recall given a recommendation set of size 10 jumps from 66.9% to 69.5%, a 2.6% increase. Boosting therefore appears to increase the completeness of the recommendation set.

Delicious					
$KNN-UT, b=0.00$			$KNN-UT, b=0.05$		
N	**Rec.**	**Prec.**	**N**	**Rec.**	**Prec.**
1	0.211	0.567	1	0.232	0.606
2	0.332	0.489	2	0.361	0.519
3	0.418	0.435	3	0.450	0.459
4	0.485	0.394	4	0.516	0.413
5	0.538	0.361	5	0.568	0.376
6	0.580	0.333	6	0.609	0.345
7	0.613	0.307	7	0.641	0.317
8	0.636	0.282	8	0.664	0.291
9	0.653	0.259	9	0.682	0.268
10	0.669	0.239	10	0.695	0.248
Citeulike					
$KNN-UT, b=0.00$			$KNN-UT, b=0.05$		
1	0.201	0.404	1	0.255	0.509
2	0.292	0.309	2	0.355	0.377
3	0.346	0.252	3	0.407	0.299
4	0.383	0.213	4	0.439	0.247
5	0.407	0.184	5	0.456	0.208
6	0.427	0.162	6	0.465	0.178
7	0.444	0.145	7	0.474	0.157
8	0.457	0.131	8	0.479	0.139
9	0.467	0.120	9	0.484	0.125
10	0.474	0.110	10	0.488	0.113

Table 4: The recall and precision for the top 10 recommended tags of K-Nearest Neighbor applied to the Delicious and Citeulike datasets. Users are modeled as vectors over the tag space. Vector weights are computed as the tag frequency. k was set to 50. N is number of tags recommended. Detailed results for two different boost values, 0.00 and 0.05, are presented.

This behavior is consistent with all K-Nearest Neighbor experiments conducted on the Delicious dataset and across all user models and all values for k; boosting results in an approximate 2.0% to 4.0% increase in recall and precision. The optimum value for b is between 0.05 and 0.075. For consistency all other K-Nearest Neighbor experiments are run using a boosting factor of 0.05.

Similar results were discovered using Citeulike data. Precision for a recommendation set of size 1 jumps from 40.4% to 50.9%, a dramatic increase of 10.5%. However, this improvement diminishes as the size of the recommendation is increased. Precision for a recommendation set of 5 climbs from 18.4% to 20.8%, a 2.4% improvement. For a recommendation set of size 10, the improvement shrinks to 0.3%.

An examination of recall shows similar signs. For a recommendation set of size 1, the improvement is 5.4%. The improvement drops to 4.7% for a recommendation set of size 5 and drops further to an improvement of 1.4% when N is increased to 10.

In general boosting tags based upon previous usage is demonstrated to add additional utility to the K-Nearest Neighbor algorithm. Yet differences in the improvements between Delicious and Citeulike offer additional insights.

First the average number of tags per posts in Delicious is 3.82. Citeulike has less with 2.5 tags per post, only 65% the number found in Delicious. As a result Citeulike presents a smaller target for tag recommendation. Moreover since the holdout set is smaller for Citeulike, boosting will have a greater impact on smaller recommendation sets than on larger sets when contrasted with Delicious. This is observed as the improvement to precision garnered from boosting drops from 10.5% when the recommendation set contains 1 tag to only 0.3% when the recommendation set consists of 10 tags. Improvements in Delicious, on the other hand, are more stable dropping from 3.9% to 1.8%. An examination of recall shows a parallel trend; in both cases recall improves as the size of the recommendation set increases subject to diminishing returns. However the effect of diminishing returns hampers Citeulike recommendations earlier and more strongly than it affects Delicious. These observations suggest that care must be taken when comparing recommendation algorithms across multiple Folksonomies.

Furthermore the two datasets have a markedly different focus. Delicious users are able to annotate any URL on the Web often tagging resources across many different topics. Citeulike users on the other hand are focused on scholarly publications and often focus primarily on their area of expertise. This observation suggests that recommendation algorithms giving added weight to resources may be more appropriate for Delicious since user information may befuddle the recommendation algorithm by incorporating unrelated tags. Conversely if a user in Citeulike has annotated items related to his research and does not stray from this topic, the user profile based on tags could offer exceptional utility.

Evidence for this analysis is provided in the difference of precision and recall between the two datasets. For a recommendation set of size 1, boosting a user's previously assigned tags offers a 3.9% gain in Delicious and a 10.5% gain in Citeulike. This trend continues as N increases until the improvements gradually diminish. This stark contrast suggests that recommendation algorithms augmented by boosting tags offer more gain for focused Folksonomies such as Citeulike than broader Folksonomies such as Delicious.

Having ascertained the optimal values for k and the boosting factor we turn our attention to the user models for K-Nearest Neighbor. Experimental results are shown in Figure 3 detailing the recall and precision for recommendation

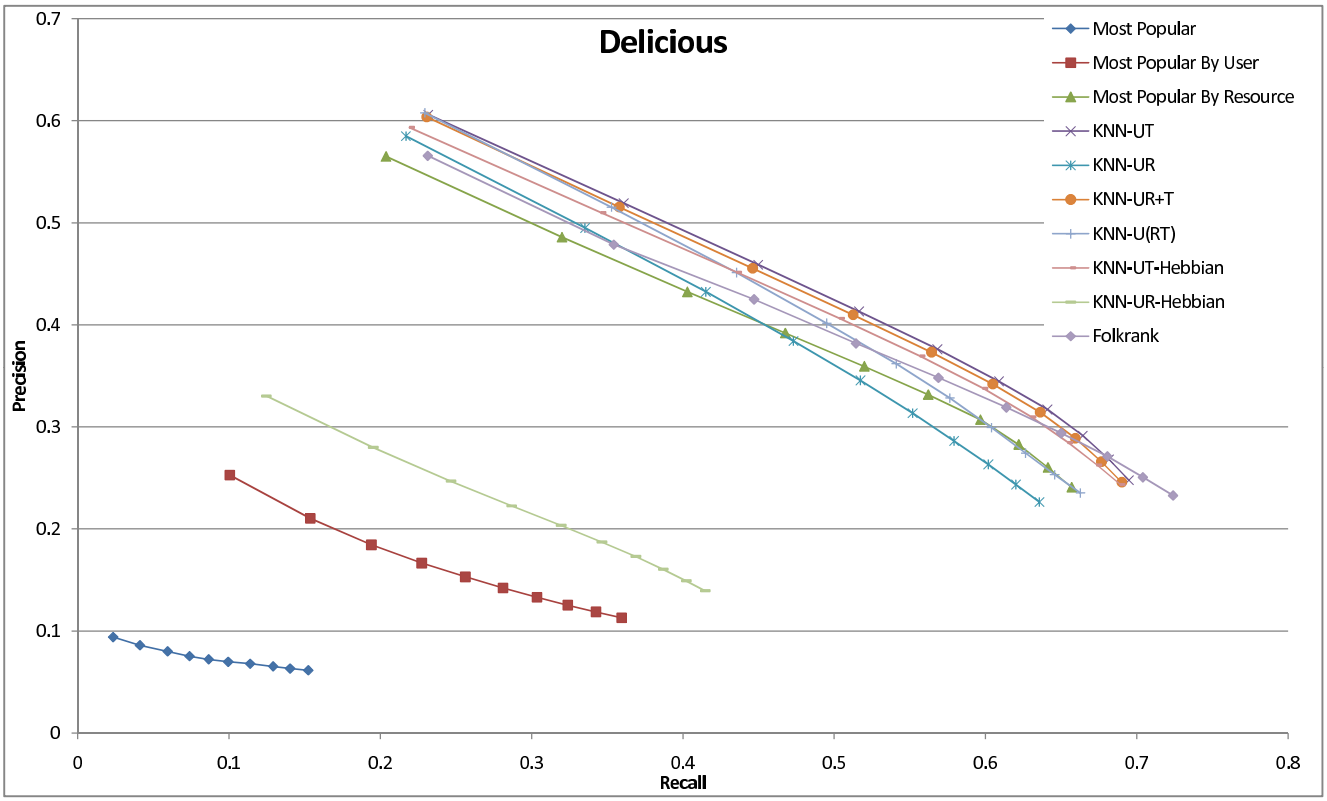


Figure 3: The comparison of tag recommender strategies for Delicious.

sets of size 1 through 10. For comparison purposes recommendation algorithms based on popularity are included. "Most Popular" recommends the most popular tags in the dataset. "Most popular by User" recommends the most popular tag for a particular user. "Most popular by Resource" recommends the most popular tags for a given resource. K-Nearest Neighbor models users as a vectors over the tag space (KNN-UT), vectors over the resource space (KNN-UR), a concatenation of the two (KNN-UR+T), a strict combination of every resource-tag pair (KNN-U(RT)) and as features derived through Hebbian deflation on either the *UT* or *UR* matrix (KNN-UT-Hebbian or KNN-UT-Hebbian). For all K-Nearest Neighbor models, k is set to 50 and the boost factor is set to .05. "Folkrank" is provided for further comparison and adapts link analysis to the folksonomy structure, recommending tags through manipulation of the preference vector.

The approach that merely recommends tags that are popular throughout the Folksonomy achieves poor results in the Delicious dataset. Recommending popular tags for a specific user fairs better, but is clearly out done by recommending popular tags given a specific resource. Nearly all user models for K-Nearest Neighbor surpass these techniques based upon popularity. In particular models that treat each user as a vector over the set of tags appear to perform best for the Delicious dataset. Folkrank offers additional completeness as seen by its superior recall, but offers less specificity as measured by it precision.

Similar trends are observed for Citeulike except for a few notable exceptions. First recommendations that rely on the popularity of a tag given a user outperform recommendations based on the popularity of a resource. This reaffirms our notion that the focused nature of Citeulike is better suited for algorithms that rely on user-tag information whereas resource-tag information is critical in broader Folksonomies where a user's annotations cover multiple topic areas.

Folkrank outperforms other methods as a measure of recall to a larger extent than in the Delicious dataset, but further trails as a measure of precision.

As in Delicious, K-Nearest Neighbor which treats users as vectors over the set of tags performs strongly in Citeulike, whereas the effectiveness of other approaches vary. The ability of this model to outperform other methods in both datasets should be noted. This is due to the inherent comprehensiveness of the *KNN-UT* algorithm.

Only those users that have tagged the query resource are considered for the neighborhood resulting in an algorithm that focuses on user-resource information. Then only those tags that have been applied to the query resource are considered for the recommendation set focusing on the resource-tag connections. Finally by treating user models as vectors over the tag space the recommender incorporates user-tag relationships. Consequently the *KNN-UT* algorithm accounts for all three aspects of the Folksonomy and is adaptable to many Folksonomies that may require an emphasis on spe-

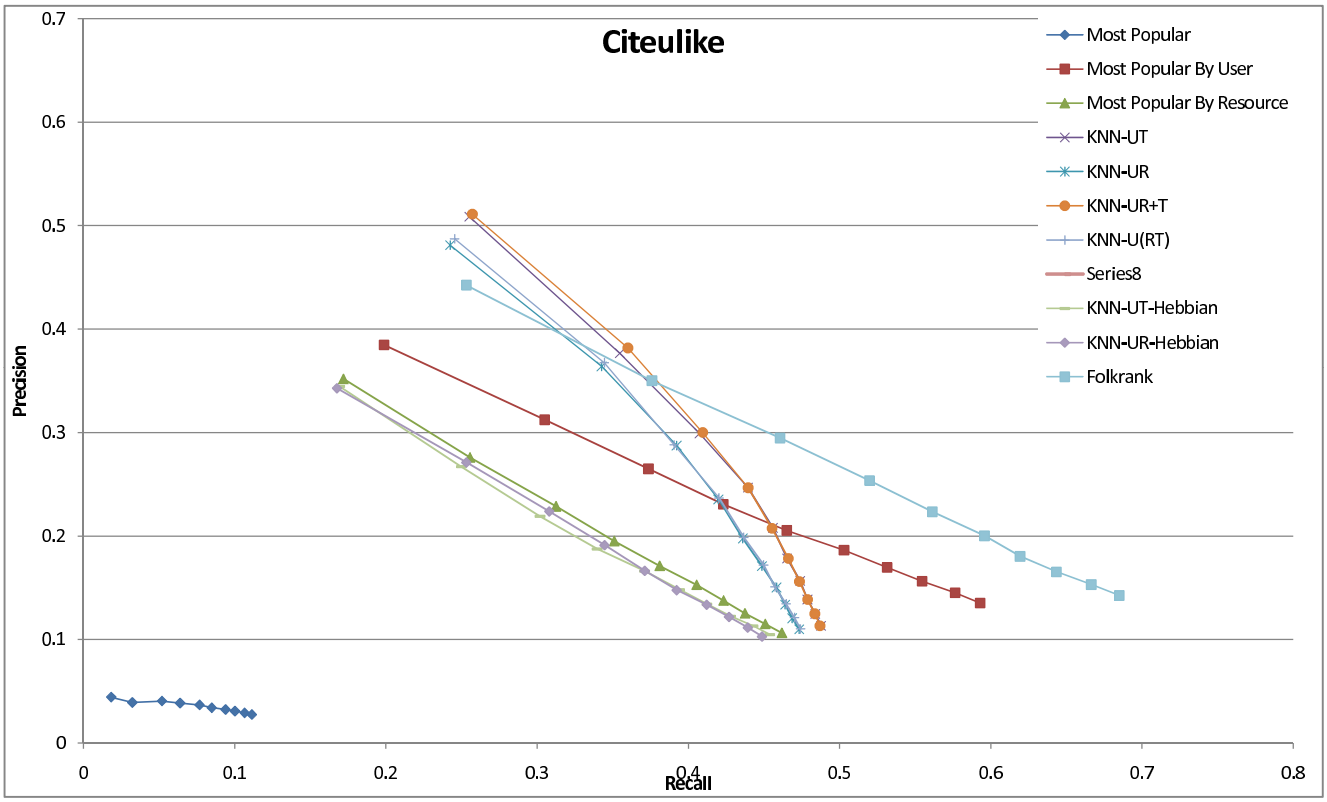


Figure 4: The comparison of tag recommender strategies for Citeulike.

cific relationships. Not surprisingly the user models that perform nearly as well as *KNN-UT* are *KNN-UR+T* and *KNN-UT-Hebbian* which share the same characteristic. Though *KNN-UT-Hebbian* does perform relatively poorly in Citeulike, likely due to the fact that this dataset is sparse and the Hebbian features are more difficult to extract.

Beyond recall and precision, time constraints should be considered when evaluating tag recommendation algorithms. Recommenders based on popularity can perform much of the computation offline thereby streamlining the recommendation process. K-Nearest Neighbor on the other hand is often referred to as a lazy algorithm since it performs the bulk of its computation during the query process. However since the proposed modifications to the algorithm limit the number of similarities that must be calculated to only those users that have tagged the query resource, the algorithm scales very well with large datasets. The computational cost may be reduced further if Hebbian features are extracted from the Folksonomy thereby reducing the length of vectors used for calculating cosine similarity. Hebbian deflation however is appropriate only when the data is dense enough that meaningful features can be extracted.

Folkrank, while it performs well in tag recommendation, is hampered by computational costs requiring a complete calculation of the Pagerank vector for each query. For example to compute 18,105 recommendations required 80 hours of computation on a modern dual-core desktop. In contrast 2,309,427 recommendations were completed in less than 1 hour using K-Nearest Neighbor.

Conclusions and Future Work

In this work we have proposed using K-Nearest Neighbor for tag recommendation in Folksonomies. Due to the unique data structure of Folksonomies, modifications are required to adapt the algorithm. Neighbors are selected only if they have tagged the query resource and tags are selected for the recommendation set only if they have been applied by the neighbor to the query resource. These modifications tie user-resource and resource-tag information into the algorithm while it dramatically reduces the computational costs. There exists a myriad of ways in which to calculate user similarity; We have found that cosine similarity between users modeled as vectors over the tag space performs well. This model incorporates user-tag information into the algorithm. By including all three relationships inherent in Folksonomies, the algorithm is robust for both broad and narrow Folksonomies. In addition, K-Nearest Neighbor can be improved by boosting tags the user has previously used. The performance of K-Nearest Neighbor exceeds that of recommendation algorithms based on popularity, while the running time makes it computationally viable for large real world Folksonomies.

In the future we plan to investigate alternative tag recommendation strategies and study resource or user recommendation algorithms. Other approaches such as association rules mining and neural networks are worth considering

for recommendation in Folksonomies. Probabilistic Latent Semantic Analysis offers an alternative means to derive features from the Folksonomy. Feature extraction of any sort presents intriguing opportunities in search, navigation, personalization and recommendation.

Acknowledgments

This work was supported in part by the National Science Foundation Cyber Trust program under Grant IIS-0430303 and a grant from the Department of Education, Graduate Assistance in the Area of National Need, P200A070536.

References

Adrian, B.; Sauermann, L.; and Roth-Berghofer, T. 2007. Contag: A semantic tag recommendation system. In Pellegrini, T., and Schaffert, S., eds., *Proceedings of I-Semantics' 07*, pp. 297–304. JUCS.

Basile, P.; Gendarmi, D.; Lanubile, F.; and Semeraro, G. 2007. Recommending smart tags in a social bookmarking system. In *Bridging the Gep between Semantic Web and Web 2.0 (SemNet 2007)*, 22–29.

Batagelj, V., and Zaveršnik, M. 2002. Generalized cores. *Arxiv preprint cs/0202039*.

Golder, S. A., and Huberman, B. A. 2006. Usage patterns of collaborative tagging systems. *Journal of Information Science* 32(2):198.

Heymann, P.; Ramage, D.; and Garcia-Molina, H. 2008. Social tag prediction. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, 531–538. New York, NY, USA: ACM.

Hotho, A.; Jaschke, R.; Schmitz, C.; and Stumme, G. 2006. Information retrieval in folksonomies: Search and ranking. *The Semantic Web: Research and Applications* 4011:411–426.

Jaschke, R.; Marinho, L.; Hotho, A.; Schmidt-Thieme, L.; and Stumme, G. 2007. Tag Recommendations in Folksonomies. *LECTURE NOTES IN COMPUTER SCIENCE* 4702:506.

Lipczak, M. 2008. Tag recommendation for folksonomies oriented towards individual users. In *Proceedings of the ECML/PKDD 2008 Discovery Challenge Workshop, part of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*.

Macgregor, G., and McCulloch, E. 2006. Collaborative tagging as a knowledge organisation and resource discovery tool. *Library Review* 55(5):291–300.

Mathes, A. 2004. Folksonomies-Cooperative Classification and Communication Through Shared Metadata. *Computer Mediated Communication, (Doctoral Seminar), Graduate School of Library and Information Science, University of Illinois Urbana-Champaign, December*.

Mika, P. 2007. Ontologies are us: A unified model of social networks and semantics. *Web Semantics: Science, Services and Agents on the World Wide Web* 5(1):5–15.

Nakamoto, R. Y.; Nakajima, S.; Miyazaki, J.; Uemura, S.; Kato, H.; and Inagaki, Y. 2008a. Reasonable tag-based collaborative filtering for social tagging systems. In *WICOW '08: Proceeding of the 2nd ACM workshop on Information credibility on the web*, 11–18. New York, NY, USA: ACM.

Nakamoto, R. Y.; Nakajima, S.; Miyazaki, J.; Uemura, S.; and Kato, H. 2008b. Investigation of the effectiveness of tag-based contextual collaborative filtering in website recommendation. In *Advances in Communication Systems and Electrical Engineering*, 309–318. Springerlink.

Oja, E., and Karhunen, J. 1985. On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *Journal of Mathematical Analysis and Applications* 106(1):69–84.

Salton, G., and Buckley, C. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing and Management: an International Journal* 24(5):513–523.

Salton, G.; Wong, A.; and Yang, C. 1975. A vector space model for automatic indexing. *Communications of the ACM* 18(11):613–620.

Schmitz, C.; Hotho, A.; Jaschke, R.; and Stumme, G. 2006. Mining association rules in folksonomies. In *Proc. IFCS 2006 Conference*, 261–270. Springer.

Sigurbjörnsson, B., and van Zwol, R. 2008. Flickr tag recommendation based on collective knowledge. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, 327–336. New York, NY, USA: ACM.

Song, Y.; Zhuang, Z.; Li, H.; Zhao, Q.; Li, J.; Lee, W.-C.; and Giles, C. L. 2008. Real-time automatic tag recommendation. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, 515–522. New York, NY, USA: ACM.

Tso-Sutter, K. H. L.; Marinho, L. B.; and Schmidt-Thieme, L. 2008. Tag-aware recommender systems by fusion of collaborative filtering algorithms. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, 1995–1999. New York, NY, USA: ACM.

Van Rijsbergen, C. 1979. *Information Retrieval*. Butterworth-Heinemann Newton, MA, USA.

Xu, Z.; Fu, Y.; Mao, J.; and Su, D. 2006. Towards the semantic web: Collaborative tag suggestions. *Collaborative Web Tagging Workshop at WWW2006, Edinburgh, Scotland, May*.

Analysis of Web Usage Patterns in Consideration of Various Contextual Factors

Jinhyuk Choi
Korea Advanced Institute of Science and Technology (KAIST)
119, Munjiro, Yuseong-gu
Daejeon, 305-732, Republic of Korea
demon@kaist.ac.kr

Jeongseok Seo
Information and Communications University (ICU)
119, Munjiro, Yuseong-gu
Daejeon, 305-732, Republic of Korea
chaoticblue1@icu.ac.kr

Geehyuk Lee
Korea Advanced Institute of Science and Technology (KAIST)
119, Munjiro, Yuseong-gu
Daejeon, 305-732, Republic of Korea
geehyuk@kaist.ac.kr

Abstract

It is important to analyze user's Web usage logs for developing personalized Web services. However, there are several inherent difficulties in analyzing usage logs because the kinds of available logs are very limited and the logs show uncertain patterns due to the influences of various contextual factors. Therefore, speculating that it is necessary to find what contextual factors exert influences on the usage logs prior to designing personalized services, we conducted several experiments in-series not only in situations of performing designed tasks during short time periods but also in users' natural Web environments during a period of several days. From the results of our experiments, we found that interest levels, credibility levels, page types, task types, and languages are influential contextual factors in a natural Web environment. Moreover, some historical and experiential patterns that could not be observed in short time analysis were discovered in the results of long time analysis. These findings will be useful for other researchers, practitioners, and especially for developers of adaptive personalization services.

Introduction

The World Wide Web has a unique characteristic in that the amount of contained information is continuously increasing and yet can still be reached easily by users through various Web services. Moreover, it provides various types of media so that users can use it for multiple purposes. Therefore, it is very important for researchers and practitioners to make the Web even more effective for finding necessary information.

One of various means by which we can make the Web more useful is to develop intelligent information delivery in order to allow users to find their target information more effectively. A core part of intelligent information delivery is to search through personalized contents without the user's explicit participation. For personalization, it is necessary to learn more about the user and to build a user model based on this knowledge. This personalization process is a main topic of research on Web usage mining (Mobasher et al. 2000; Gauch et al. 2007). However, it is not easy to learn more user information because we cannot explicitly ask the user about his/her characteristics or what he/she is thinking at any particular time we want to know. This means that we have to find another way to learn more information about them. From this perspective, many researchers have looked for effective implicit methods to learn more about users, and many intelligent methods have been actively suggested by several researchers (Kelly and Teevan 2003; Kelly 2004; Kelly and Belkin 2004; Kelly and Cool 2002; Choi et al. 2007; Hofgesang 2006; Seo and Zhang 2000; Badi et al. 2006; Al halabi et al. 2007; Kellar et al. 2005). In their researches, usage logs that are stored while users visit Web pages have been used to learn about particular user interests. For examples, the URLs of visited Web pages, visit period, dwelling time, mouse clicks, mouse movement, keyboard typing, and visit frequencies on each Web page have been applied as implicit interest indicators.

Although many successful results have been provided so far, there are several inherent difficulties in analyzing usage logs and extracting necessary information from them. The first difficulty comes from the fact that the kinds of available usage logs are very limited, and there are no standard ways to interpret the meaning of usage patterns. This means that we have to carefully investigate usage patterns prior to using the logs as effective indicators. Secondly, Web users are under the influence of various contextual factors while they use the Web, as it has multiple aspects as a simple information tool, social communication mediator, entertainment source, and so on. Therefore, usage logs will show very uncertain patterns because various contextual factors will exert their influence on the usage patterns concurrently (Kelly and Belkin 2004). The third difficulty is related with the historical aspect in that a user's experiences also exert influences on the variation of usage patterns. Therefore, a Web usage pattern analysis should be a long-term process

because it cannot be adequately performed by studying only short-time usage. In addition, to analyze a user's various characteristics, the usage data should be collected at the browser side in the user's real Web environment for a long period without any constraint on a specific Web server.

This paper details the results of our experiments in which we initially tried to find the possibilities of overcoming the above difficulties. For our experiments, various usage logs have been collected at the browser side and carefully analyzed not only in situations of performing designed tasks during short time periods but also in users' natural Web environments during a period of several days. We obtained several interesting findings from the results. We think that these findings will be useful for other researchers, practitioners, and especially for developers of personalization services.

This paper is organized as follows. In section 2, we review some of the related researches. In section 3, we describe our experimental procedure and the results that have been obtained so far are given in section 4. In section 5, summary and future works are introduced.

Related Work

Human Information Behavior

There have been a lot of studies that have focused on human information behavior analyses in various research fields. In those studies, the researchers have focused on several contextual factors that affect a user's behavior, conceptualizing the relationships between information-seeking behavior and contextual factors. In (Sonnenwald 1999), the authors proposed an evolving framework in which cognitive, social, and system perspectives are incorporated. In the framework, human information behavior including information exploration, seeking, filtering, use, and communication were included. Based on the framework, various influential factors - physical, cognitive, affective, economic, social, and political – and their implications were investigated. In (Johnson 2003), the needs of an information-seeking behavior analysis in a multi-contextual environment were presented and a theoretical framework was suggested. The authors of (Kari and Savolainen 2007) asserted that users are also improving along with the change of information environment, and they found 11 relationships between individual developmental objectives and information searching via the Internet. In (Byström and Järvelin 1995; Borlund and Ingwersen 1997; Bystrom 2002; Vakkari 1999; Vakkari 2001), the influence of task complexity on information seeking behaviors was investigated. An overview of the nature of trust, and a framework of trust-inducing interface design features, were given in (Wang and Emurian 2005). Particularly in (Wang et al. 2000), the authors introduced a multidimensional model of user-web interaction, and three dimensions – user, interface, and the Web – were considered. In the model, the user dimension is considered to be influenced by the particular task, information need, knowledge state, cognitive style, affective state, and so on. They measured users' cognitive styles and affective states before a user study, applying a process-tracing technique while users were conducting information-seeking tasks, and found various types of relationships among the elements of the dimensions. In (Fogg et al. 2003), based on results of an online qualitative study, the credibility for Web contents were considered and important factors of credibility were suggested to formulate Web design guidance. In (Wathen and Burkell 2002), the authors asserted that users filter out most of the gathered information and retain only useful information. In addition, they concluded that the credibility or believability of information is one of the most important criteria for the filtering. In (Rieh 2002), the authors found that users judge cognitive authority and information quality by two types of judgment - predictive judgment and evaluative judgment – and they also identified the main facets and keywords of the judgments through a user study.

From these researches, we found that human information behavior cannot be studied without the consideration of the influences of various types of contextual factors. However, because the purposes of these researches were not to develop an intelligent system but to construct theoretical models, they did not study quantitatively how the Web usage patterns reflect the influences of various contextual factors.

Web Usage Mining

There has been a lot of effort to quantitatively measure the influences of contextual factors on Web user behaviors based on various usage logs in the field of Web usage mining. Among the various factors, user interest toward content has been the main focus of researchers. The various implicit indicators of user interest can be found in (Kelly and Teevan 2003). In (Kelly 2004), the familiarity of a topic has been discussed, and the authors concluded that as one's familiarity with a topic increases, his/her searching efficacy increases and reading time decreases. For user characteristics, cognitive and problem-solving styles were studied in (Kim and Allen 2002). In their study, the authors observed various user activities - average time spent, average number of Websites viewed, average number of bookmarks made, and average number of times a search/navigational tool was used for completing a search task – while the users performed two types of given tasks in an experimental environment, and the authors found that there are significant differences among user activities according to the type of task and user's problem solving style. For usage logs, the display time was discussed most actively. In (Kelly 2004; Kelly and Belkin 2004), based on gathered data from 7 subjects for 14 weeks, the relationships between display time and various factors – task, topic, usefulness, endurance, frequency, stage, persistence, familiarity, and retention – were investigated, and the authors concluded that the display time is not suitable for inferring a user's interest because there is great

variation between display time and interest according to the user; large differences according to the task at hand also appear. On the contrary, in (Choi et al. 2007), the viewing time has been used as a good implicit indicator, and in (Hofgesang 2006), the authors made an assertion that time spent on a Web page is more important than visit frequency in inferring a user's interest. In (Seo and Zhang 2000), bookmarking, time for reading, following up the HTML document, and scrolling were used as relevant activities, and a machine learning algorithm was applied to learn the user's characteristics. In (Badi et al. 2006), various parameters of document attributes, document reading activities, and document organizing activities were investigated to recognize user interest and document values. In (Kellar et al. 2005), the authors found that the time spent is more useful for more complex Web searching tasks. In (Nakamichi et al. 2006), the authors also used several quantitative data of user behavior – browsing time and moving distance, moving speed, and wheel rolling of the mouse – to detect low usable Web pages.

Most of the researches have analyzed usage logs with the intention of developing an intelligent system that learns user characteristics and builds a user model. However, most of the studies did not fully consider the influences of various contextual factors, or they focused only on a user's interest without consideration of other types of subjective feedback together. Moreover, most researches except (Kelly, 2004; Kelly and Belkin 2004) did not consider the historical aspects of usage data that can only be gathered by a long-time analysis in a user's natural Web environment.

Our Approach

Before everything else, we reviewed previous related researches carefully and collected contextual factors for consideration and usage logs that can be obtained at the browser side. The contextual factors and usage logs that we considered are given in Table 1.

We carried out not only a qualitative analysis but also a quantitative one. For ecological validity, we also observed users in their own personal places. Because some of the contextual factors are inherently subjective and cannot be measured with only usage logs, we collected various types of feedback regarding the current context directly from users. However, to minimize the burden on the users in this study, we tried to minimize the number of feedback questions as much as possible. We developed software that runs on each user's PC in order to collect their behavior logs and feedback in their Web browsing environments.

Contextual Factor

Contextual factors include subjective assessments about contents, situational factors, a user's individual characteristics, and so on. Because these factors cannot be measured systemically, we designed a process in which we can obtain the users' subjective feedback directly. First, we considered the users' attitudes toward the current task as one of the contextual factors. Actually, the types of user task can be classified into detailed categories – information seeking, fact-finding, transaction, and browsing (Kellar et al. 2007). However, we classified user tasks into only two categories – careful searching and casual searching - according to the users' attitudes toward the current task. A detailed description of the task categorization appears in section 3.5. There are more contextual factors that cause users to interact with Web pages. For example, a user may stay for a relatively long time at a specific Web page because there are interesting contents there, or the user feels that the contents are more useful than others. Sometimes, the user may roll the mouse wheel more frequently on one Web page than on others because he/she wants to read the entire content of the page carefully. In this regard, we selected some further factors that may exert an influence on user interactions with Web pages. The factors are interest, credibility, complexity, and difficulty. The complexity factor tells us how users feel about the layout structure of a Web page, and hence it may include a user's subjective viewpoint of usability and familiarity. We also included the difficulty factor because we thought that user behavior is subject to variation according to a subjective assessment of the difficulty of the contents displayed.

	Task	**Contextual factors**	**Usage logs**	**Period**
Ex1	Visit collected pages (text only)	Interest	Viewing time Mouse movement Mouse wheel Mouse clicks WM_PAINT	2 hrs
Ex2	Visit collected pages	Interest Complexity Difficulty Credibility	Viewing time Mouse movement Mouse wheel Mouse clicks WM_PAINT	2 hrs
Ex3	Free visits / given tasks	Interest Complexity Difficulty Credibility Task type	Viewing time Mouse movement Mouse wheel Mouse clicks WM_PAINT	2 hrs
Ex4	Free visit / free tasks	Interest Credibility Task type	Viewing time Mouse movement Mouse wheel Mouse clicks Keyboard typing Visit frequency Day frequency	2 wks

Table 1. The environment and gathered data of experiments

Web Usage Log

Implicit user interest analysis has shown good performance at the server-side especially for commercial Websites. However, in spite of the fact that it is easier to analyze user interest at the server-side, currently many researchers have focused on browser-side analyses because user interest can be analyzed from various Websites, and a user model can

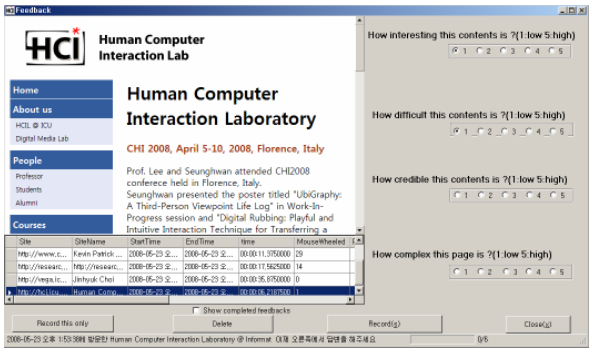


Figure 1. The feedback window consists of a browser control to view the contents of visited web pages, a list window to choose a visited URL, radio buttons to choose the answer of some questions, and so on

be constructed using a wealth of information through a browser side analysis. In order to analyze users' implicit interest at the browser side, we have to monitor several usage logs, for example, the viewing time, scroll movement, sequences of visited URLs, keyboard typing, and so on. In our research, we have chosen several usage logs to record while users view different Web pages. The viewing time that has mainly been investigated in the related researches so far is the time during which users remain on a particular web page. The mouse wheel counts the number of WM-MOUSEWHEEL messages (Choi et al. 2007). For mouse and scrollbar movement, we measured the distance between two consecutive positions of the mouse cursor and scroll bar at regular intervals and summed the distances. We also counted the number of processed WM-PAINT messages, as WM-PAINT messages are processed when users change the size of their browser window, scroll within the window, move their mouse cursor, and so on. The number of mouse clicks and keyboard typing were also considered. We believe that these activities are good indicators of user interest regarding the contents of Web pages. We have chosen these logs because they can be measured without much effort. However, for scroll movement, we were unable to obtain the position of the scrollbar on some of the Web pages, and the WM-PAINT messages can be affected by the dynamic content of certain Web pages. This means that we have to be careful when using these data as logs for measuring user activities.

We did not record some of the behaviors that have been considered by other researches – bookmarking, saving, printing, and coping and pasting – because users do not always show those behaviors on every valuable Web page, and hence their records do not suit our purpose.

We collected some physical data of Web pages - the scroll height, file size, and URL information (top-level URL and depth of URL) - of each visited Web page. Moreover, in the course of the experiments, some additional factors were included when they were required for analysis. The additional factors were the number of out-links on a Web page, the Web page type, and the language presented (e.g., Korean or English).

We also considered carefully some historical factors that can be analyzed only through relatively long periods of monitoring. The historical factors include visit frequencies and day frequency. Among those factors, day frequency is a new concept that has not been introduced before. A detailed description of day frequency will be given in a later section.

Data Collection Software

In some of the previous researches, custom-built browsers have been used (Kellar et al. 2007), as have some specialized logging software that works "in stealth mode" (Kelly and Belkin, 2004). Although there are several merits in using custom-built browsers, because various data can be collected easily, we developed a browser-monitoring module (BMM) that runs behind Internet Explorer without any modification to the browser, as we wanted to preserve the natural state of the Web browsing environment as much as possible.

BMM is a type of monitoring software that was developed to detect Windows GUI messages while users read Web pages, and thus it is possible to measure user activities in real-time without any interruption to the users. BMM uses a global hooker library, written in C++, which runs in the background and hooks all Windows operating system events. In addition, using Windows Shell API, BMM can access all instances of currently running Internet Explorers through the COM object. In addition, necessary properties of Web pages can be obtained from the COM object. BMM is written in C#, running under a Windows platform with .NET Framework 2.0.

BMM consists of four components - hooker, data recorder, data aggregator, and feedback window. The data to hook are the number of keys pressed, events of program focus changes, number of WM_PAINT events, mouse click and mouse wheel messages, and so on. Basically, the hooker catches every message passed within the operating system, so we should filter out irrelevant messages to record only necessary data for our studies. For instance, because a WM_PAINT message is invoked whenever the O/S needs to re-draw some parts of a window, we have to be able to ignore the messages from unfocused windows and count the number of messages that are invoked for only the currently focused browser window. The aggregator can acquire several properties of a Web page by using a Document Object Model (DOM). Acquired properties are the viewing size of a document (in pixels), file size (in bytes), current location of the scrollbar, and character set of the page. The location of scroll bar is periodically updated so that the total displacement of the scrollbar can be estimated. However, a critical issue arises at several 'fancy' Web pages that have different structures from standard Web documents, eventually yielding no data while accessing the DOM property. The data aggregator also aggregates all data from these multiple components, and the data recorder stores the aggregated data in a

human-readable XML format for future analysis. After Web searching, using the feedback window, users can review the visited Web pages and choose radio buttons that ask about several types of assessments about the contents of each Web page. If the users do not want to answer questions regarding some of the Web pages, they can even remove the records easily. In figure 1, the structures of the feedback windows are shown.

Subject

We conducted 4 experiments, each with its own purpose. The detailed concept of the experiments will be described in the next section. For each experiment, we recruited some graduate students who are majoring in computer science for our subjects. Twenty-five students participated in the first experiment, 23 in the second, 19 in the third, and 12 students in the fourth. Among the students, 11 got through the second, third, and fourth experiments, and one new subject volunteered for the fourth experiment. All of the students have a high level of knowledge and experience about the Internet and the Web. We chose these students as subjects because all of them use the Web not only for their work but also for entertainment or distraction. Most of all, they use the Web for a relatively long time each day so that we could gather plenty of data from their activities. It also means that we could observe their Web usage patterns under various contexts. We paid about 20 dollars to each subject for their participation in the first, second, and third experiments, respectively. For the fourth experiment, we paid 60 to 160 dollars to each subject according to the rate of the completed feedback.

Experimental Concept and Procedure

There are three main strategies for studying information-seeking behavior – laboratory experiments, sample surveys, and field studies (Kellar et al. 2007). Considering these strategies, we designed four experiments and conducted them in-series. In the first and second experiments, the subjects came to our laboratory and browsed some pre-collected Web pages. In the third experiment, the subjects performed given information-seeking tasks in our laboratory. As a final step of each experiment, the subjects carried out feedback tasks in order to record their own subjective assessments about each of the Web pages they had browsed. The fourth experiment was carried out at the subjects' own residences. The subjects installed BMM on their PCs to collect their Web usage logs for a period of about two weeks. For the feedback process of the fourth experiment, we let the subjects carry out the feedback tasks at least once a day. The first and second experiments were carried out in a blind mode in which the subjects could not see any information about the contents of each Web page before viewing them. In other words, no proximal cues (Chi et al. 2001) were provided.

Experiment 1. The first experiment was a kind of preliminary study. We collected 120 Web pages that contain only text and offer information on various topics – politics, economics, education, engineering, entertainment, science, health, and sports – with varying content size. The twenty-five subjects read each page in their own desired manner from the list of collected Web pages. Because we wanted to exclude any effect of information clues, we simply provided numbers on the list without showing any information about the contents of the Web pages in advance. Thus, the subjects were supposed to click the numbers in order to view the contents. To obtain the appropriate data, the subjects were not told that some activities would be measured while they were viewing the Web pages. During the experiments, the subjects' activities while reading the Web pages, and some measurable data, were recorded in a log file for future analysis. In addition, whenever a subject finished reading a Web page, a small window appeared wherein the subject recorded his/her interest level for the contents of the page. There were 5 levels of interest, and the subjects recorded their interest for the contents of a Web page accordingly. Due to some malfunctions of the BMM in the users' browsing environment and a failure to properly obtain user feedback, the log files of 5 users were excluded. Therefore, we analyzed 20 users' log files. For the first experiment, we formulated the following simple hypotheses.

1. The number of processed log data is relatively higher on Web pages that contain interesting contents.
2. The amount of information in a Web page affects the amount of processed log data.

Experiment 2. Actually, the procedure of the second experiment was the same as the first experiment except that we collected ordinary Web pages that contain images, tables, videos, and frames. It was intended to see whether there will be differences in usage patterns according to form of the Web page. When a subject finished reading all of the Web pages, he/she activated a feedback window wherein the subject could review all of the pages and answer some questions about each one visited. In this experiment, differently from the first experiment that collected only the interest levels for the contents, we also wanted to verify the influence of other subjective assessments of Web pages - difficulty, complexity, and credibility along with interest – on a 5-point scale. If a subject clicked one of the URLs on a visited page list in the feedback window, the contents of the Web page appeared again, and the subject could then choose his/her points for the questions regarding the subjective feedback.

Experiment 3. We can find several different categorizations of Web user behaviors in previous researches. Most recently, 4 task categories were provided in (Kellar et al. 2007) - fact finding, information gathering, just browsing, and transactions. In (White and Drucker 2007), Web users are grouped into navigators and explorers according to the level of visit variances. In consideration of these previous works, we also classify a user's Web tasks into two groups.

Ex	Feedback	VT	MM	MW	MC	WP
Ex1	interest	0.695 (**)	0.572 (**)	0.563 (**)	0.475 (**)	0.663 (**)
	scroll height	-0.006	0.006	0.261	0.008	0.059
Ex2	interest	0.771 (**)	0.545 (**)	0.686 (**)	0.559 (**)	0.507 (*)
	complexity	-0.391 (**)	-0.178	-0.148 (**)	-0.196	-0.599 (*)
	difficulty	-0.476	-0.340	-0.532	-0.418	-0.057 (**)
	credibility	0.507	0.203	0.411 (*)	0.289	0.241
	scroll height	0.074	0.016	0.167	0.001	-0.059
Ex3	interest	0.396 (*)	0.301 (*)	0.119	0.245	0.229
	complexity	-0.315 (**)	-0.129 (**)	-0.533 (**)	-0.162 (**)	0.040
	difficulty	0.307	0.307	-0.124	0.182	-0.330
	credibility	0.609 (**)	0.414 (**)	0.288 (*)	0.412 (**)	0.389
	scroll height	0.011	-0.025	0.120	-0.036	-0.022
Ex4	interest	0.442 (**)	0.315	0.258	0.306	0.282 (KT)
	credibility	0.434 (*)	0.138	-0.010	0.222	0.124 (KT)
	scroll height	0.056	0.001	0.117	0.017	0.001 (KT)

VT: Viewing time / MM: Mouse move / MW: Mouse wheel / MC: Mouse click / WP : WM_PAINT / KT: Keyboard typing
*: p-value of ANOVA test < 0.05
**: p-value of ANOVA test < 0.01

Table 2. The values of correlation between feedback level and the amount of usage logs

Task 1: careful searching

This task is a type of information gathering that requires accuracy, trust, efficiency, and responsibility of the search results. In our experiment, the given task was to find some information about their research topics. For examples, they had to find some Web pages of laboratories in universities or companies that are related with their research topics and read the pages carefully to judge the relevance of the information. We encouraged the subjects to perform this task as normally as possible.

Task 2: casual searching

This task is a type of information gathering and browsing that can be performed without any burden or responsibility regarding the search results. For example, the subjects could search for some information about their hobbies, favorite products to buy, famous tourist spots, favorite sports or movie stars, and so on. We also encouraged the subjects to perform these tasks as normally as possible.

The subjects performed the two tasks with their own topics for about 2 hours. The logging data and feedback

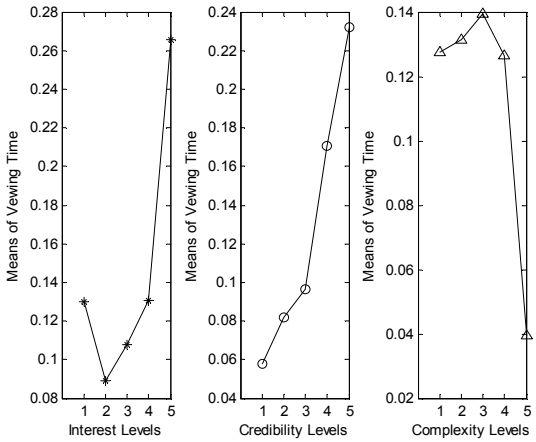


Figure 2. The viewing time according to feedback levels in the third experiment

details were the same as in the second experiment. Differently with the first and second experiments that controlled the subjects' activities in that the subjects could only visit the collected Web pages without any pre-information clues, in the third experiment, the subjects could visit any Web page that they wanted and use any search engine or portal site they wanted to use. Therefore, we observed a lot of re-visitation patterns. Thus, during the feedback phase, we let the subjects delete the logs of Web pages that they just used to find other Web pages to visit. In this way, we excluded the navigational Web pages. The concepts of the navigational Web pages will be given in section 4.5.

Experiment 4. For the fourth experiment, 12 graduate students participated - 4 females and 8 males. They installed the BMM on their PCs and collected various logs for about 2 weeks. Some of the subjects participated in our experiment for 16 days. For their feedback, we encouraged them to give their feedback levels of each visited Web page a 5-point scale and choose one of the task types. If a URL was not a content page according to the subject's viewpoint, the URL could be deleted easily and BMM records a special number for the URL for future analysis. In this experiment, we collected only three types of feedback – interest, credibility, and task types - because we wanted to minimize the subjects' burden in answering many questions for all of the visited Web pages visited.

Result

In the series of experiments, we measured the numbers of several processed messages on each visited Web page and normalized the value using min-max normalization according to each subject. We included this normalization procedure because there would be variances in the amount of usage logs due to the subjects' individual differences.

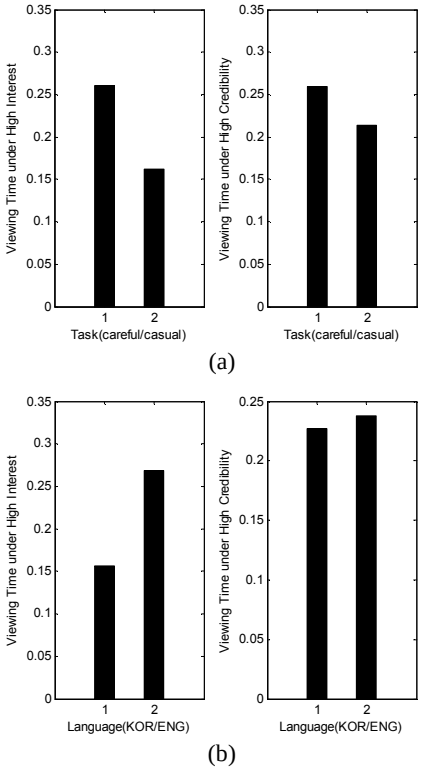


Figure 3. (a) The differences of viewing time according to task types (b) the differences of viewing time according to languages

Experiment 1

From the results of the first experiment, we found some interesting patterns. As we can see in table 2, there were positive correlations between the amount of all usage logs and interest levels. Furthermore, from a one-way ANOVA test, we also found that the amount of the logs shows significant differences among the interest levels. Based on this result, we temporally concluded that users have a tendency to interact more at high-interested Web pages, and hence all the logs can be used as implicit interest indicators. One more interest thing is that there was a low correlation between the amount of usage logs and the size of the Web pages except for the amount of the mouse wheel log.

Experiment 2

Actually, we thought that there would be some differences between the result patterns of the first experiment and those of the second experiment because the forms of the Web pages were quite different. However, there were no big differences between the results. Table 2 shows us that there were also positive correlations between the amount of all usage logs and interest levels, similarly with the results of the first experiment. In addition, we also found significant differences in the amount of usage logs among the interest levels. This means that the form of the Web pages is not an important factor. Differently from the results of interest level, the difficulty and complexity levels showed a negative correlation with the amount of usage logs. The credibility levels showed positive correlations with the amount of usage logs but the differences of the amounts among the levels are not statistically significant. From the results, we concluded that the interest level exerts the most significant influence on the amount of usage logs, and that users are inclined to quickly leave Web pages that have difficult contents or complex structures without many interactions. Finally, we found that there were low correlations between the amount of usage logs and the sizes of Web pages except for the amount of the mouse wheel log. This was not different with the results of the first experiment.

Experiment 3

In figure 2 and table 2, we can see that the viewing time and amount of mouse movement have positive correlations with the interest levels, and that the differences of the amounts among the interest levels are also statistically significant. The amount of mouse wheel use, mouse clicks, and processed WM_PAINT messages also showed positive correlations with interest levels, but the differences were not statistically significant. The amount of usage logs increased according to the complexity levels, but dropped steeply at level 5. The difficulty levels showed no large correlation with the amount of usage logs. The most interesting pattern that we found in the results of the third experiment was that the amount of usage logs showed a positive correlation with the credibility levels, and that the differences of the amounts of usage logs among the credibility levels were statistically significant. This result was not found in the results of the second experiment in which users browsed pre-collected Web pages without proximal cues. Therefore, we concluded that the usage logs are under the influence of credibility levels as well as interest levels in ordinary Web browsing environments.

In the third experiment, we also checked whether there are differences in the amount of usage logs according to the task types and written languages used. From figure 3, we found that there was a general trend of more interaction logs recorded during a careful task than during a casual one, especially on pages of the highest interest and credibility levels. For written languages, there was a general trend of more interaction logs on English pages than on Korean pages, especially on the pages with the highest interest levels, but there was no large difference according to credibility levels. These results showed us that the type of task and written languages used also should be considered as important influential factors that make differences in the amount of usage logs created.

Experiment 4

In the fourth experiment, there were some logs that contain an excessively long viewing time because the experiment has been conducted in the users' personal environments.

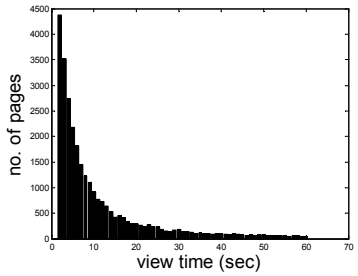


Figure 4. The distribution of viewing time

From figure 4, we can find that the users stayed on 99% of the all visited Web pages for at most 346 seconds. We also found that there were some visited logs that showed a viewing time of over 30 minutes. This means that we should find a maximum cutline in order to filter out some logs as simply noise. We set various values to the cutline, from 3 to 25 minutes. As we set the cutline values differently, we excluded logs in which the viewing time was above the cutline, and then normalized each user's viewing time to his/her scale. Finally, we checked whether the magnitude of the cutline made an impact on the applicability of the viewing time as an indicator. From figure 5, we can see that a reasonable cutline should be set to somewhere between 5 and 18 minutes in order to observe a high positive correlation between the viewing time and interest level, and the statistical difference among the viewing times in each interest level. For example, if we set the maximum cutline to 14 minutes, the viewing time shows a positive correlation with the interest level ($r = 0.5522$), and according to the result of a one-way ANOVA test, the differences among the viewing times of each level are significantly different ($p = 0.0092$). This means that we can use viewing time to identify interested Web pages based on the fact that users will stay for a relatively longer time on them than on uninterested Web pages. In addition, we found that when we want to infer users' interest based on the viewing time, a careful noise-filtering task is absolutely required. Therefore, we excluded logs that contained over 15 minutes of viewing time in the fourth experiment. In figure 6 and table 2, we can see that only the viewing time showed positive correlations and statistically significant differences among the levels of interest and credibility. It is very interesting that we could not find significant differences between other usage logs and feedback levels. The differences in the amount of usage logs according to the task types were similar with the result of the third experiment.

Additional Findings from Experiment 4

Because the fourth experiment was conducted during a period of about 2 weeks, we can observe some more historical patterns that could not be observed in previous experiments. In this section, we introduce some additional findings.

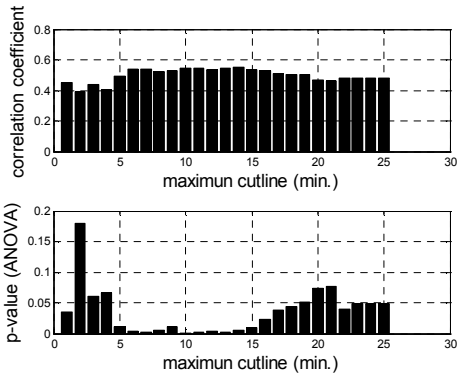


Figure 5. The correlation coefficients (top) and p-values of significance test (bottom) according to maximum cutline

Day frequency and feedback pages vs. non-feedback pages. Because most of target pages that users want to access can be reached via portal sites, news sites, and search engines, we thought that the front pages of these sites and hub pages within the sites may appear in the visited URL history more frequently than others. For example, when a user wants to read a newspaper, he/she visits the home page of news site and clicks on some links that seem to contain interesting news. In a similar manner, whenever a user wants to find some information, he/she may visit the front page of a search engine first and then click on one of the links that the search engine retrieves. Similarly, if the user wants to log onto some commercial sites or even his/her own Web mail accounts, he/she should first visit the front page of the service and input his/her username and password in order to proceed. Therefore, if we look over the users' visited URL histories, the navigational pages - the front pages of portal sites, news sites and search engines, and any type of hub page – will appear more frequently than others. Moreover, if the users visit Websites according to their daily routine, they will visit some of the Websites everyday in their regular patterns. In this respect, we thought that the URLs of navigational pages might be found in logs from each day. On the contrary, the content pages were shown relatively

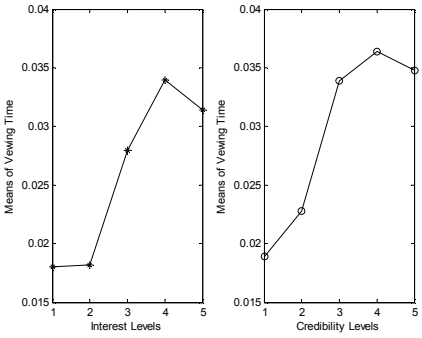


Figure 6. The viewing time according to feedback levels in the fourth experiment

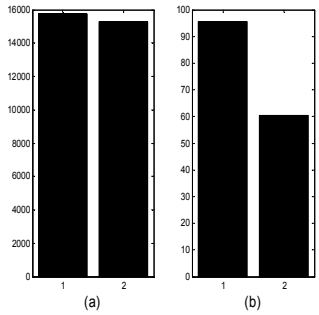


Figure 7. (a) The number of feedback pages and non-feedback pages and (b) the average number of outlinks contains: 1 – feedback page / 2- non-feedback page

rarely because the users don't usually view the same contents again and again.

Based on the considerations that we have mentioned so far, we formulated a very simple hypothesis - everyday-visited URLs have a strong chance to be navigational pages. For the hypothesis, we created a variable named Day Frequency (DF). The concept of DF is very similar to document frequency, which is often used in information retrieval and text mining (Salton and McGill 1986), and DF value of each visited URL can be calculated using equation (1).

$$DF_i = \frac{|\{d_j : Url_i \in d_j\}|}{|D|} \quad (1)$$

In this equation, $|D|$ is total number of days in experiment, d_j is the URL collection of the j-th day and $|\{d_j : Url_i \in d_j\}|$ means the number of days where i-th URL appears. If a URL exhibits a high value of DF, the URL is thought to be inappropriate for content extraction and should be regarded as a navigational page.

In the fourth experiment, the selection of a contents page was fully up to the subject's subjective decision. Even though we did not explain the concept of navigational pages in detail, they found by themselves that there are naturally several Web pages that may not be fit for expressing their feedback levels. As we can see in figure 7, the number of non-feedback pages was much greater than

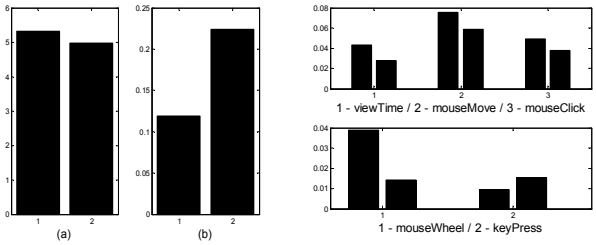


Figure 8. The URL depth of feedback pages and non-feedback pages (left - a) and the DF values (left - b) : 1 – feedback page / 2- non-feedback page and the mean values of interaction logs: on feedback pages (right - left bars) and on non-feedback pages (right - right bars)

logs	p-value
URL Depth	0.0623
Day Frequency	0.0003 (*)
Viewing Time	0.0206 (*)
Mouse Move	0.5314
Mouse Click	0.5258
Mouse Wheel	0.0181 (*)
Keyboard typing	0.0349 (*)

Table 3. The results of significance test for difference of the values of each interaction log between feedback pages and non-feedback pages: (*) means significant

our expectation. The 12 subjects have mainly deleted the home pages of search engines, retrieved lists of search engines, the first pages of portal sites, news lists, home pages of community sites, online banking sites, intranet front pages and so on as non-feedback pages. In some of the previous researches, we found that there were several attempts to discriminate content pages from navigational pages using the number of outlinks that are contained in the pages (Cooley et al. 1999; Fu et al. 2001; Domenech and Lorenzo 2007). The main idea is that there will be a larger number of outlinks on navigational pages than on contents pages. We also thought that this idea is acceptable so we counted the average numbers of contained outlinks in both feedback pages and non-feedback pages. However, as we can see in figure 7, the number of outlinks on feedback pages was higher than on non-feedback pages in our results. Therefore, we examined carefully whether the DF values in feedback pages and non-feedback pages are significantly different. As we can see in figure 8, the average DF value of non-feedback pages is higher than the values of feedback pages, and the difference is statistically significant ($p = 0.0003$). We found that the amount of some usage logs was also different between feedback and non-feedback pages. From table 3, we can see that viewing time, the amount of mouse wheel use, and the amount of keyboard typing were significantly different.

Task Identification by Visited URLs. We believed that users have their own URL lists that are specific to their current tasks because they may use the Web based on their individual previous experiences on the Web. In this respect, we analyzed the top-level URLs that users visited during the period of the experiment. As we can see in table 4, over 90% of visited URLs were separable by the tasks.

user No.	task separable (%)
1	92.68
2	92.59
3	93.17
4	95.77
5	75
6	93.86
7	100
8	90.57
9	89.29
10	97.40
11	91.07
12	96.21

Table 4. The proportion of task separable URLs

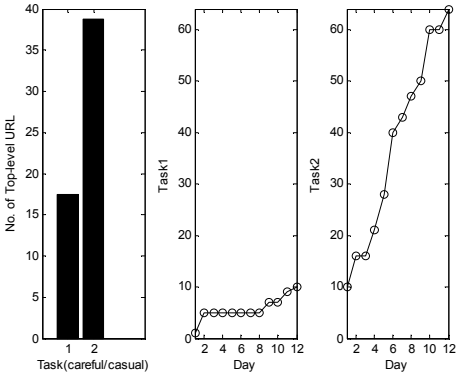


Figure 9. The average number of URLs in each task (left), the increasing rate of average number of URLs in careful task (middle), in casual task (right)

In other words, 90% of visited URLs belong to a specific task only, and hence we can infer the types of current task easily by checking the top-level URLs. Moreover, as we can see in figure 9, the number of URLs that users visited in the casual tasks is much higher than in the careful tasks. The most interesting patterns are the increasing rates of the number of visited URLs as time goes on. The number of visited URLs in the tasks of casual searching increased more drastically than in careful searching. This means that the subjects showed the navigator's patterns in careful searching tasks but showed the explorer's patterns in casual searching tasks (White and Drucker 2007). We believe that this pattern is meaningful in developing personalization schemes that are adaptive to current task types.

Discussion and Future Work

Review of the Result and Summary

We analyzed the results of 4 experiments and recognized that there are noticeable differences in usage patterns according to the experimental environment. In this section, we briefly summarize the interesting differences.

Experiment 1 vs. Experiment 2. The forms of the Web pages that the subjects visited in the first and second experiments were different, but we could not see large differences between the results of the two experiments. Moreover, the amount of usage logs was not influenced by the amount of contents or size of the Web pages. We believe that this pattern came from the fact that Web users read Web pages in a nonlinear pattern, and that there are some unique characteristics in reading digital documents (Liu 2005).

Experiment 2 vs. Experiment 3. In the results of the third experiment in which the subjects freely select the Web pages to visit, we observed that the credibility levels regarding the contents exert a noticeable influence on the amount of usage logs, but the same pattern has not been observed in the results of the second experiment in which the subjects visited pre-collected Web pages even without any pre-clue about the contents. In addition, there were significant differences among the amounts of all usage logs according to interest levels in the results of the second experiment, but only the amount of viewing time and mouse movements were affected by the interest levels in the results of the third experiment.

Experiment 3 vs. Experiment 4. Differently from the results of the third experiment, we observed that the viewing time only showed a significant relation with the interest and credibility levels in the results of the fourth experiment. This means that the more natural the environment is, the more unknown factors will exert their influences on the usage patterns. We also observed in the results of the third experiment that there are some differences in usage patterns according to the task types, such that the amount of usage logs on interested Web pages in careful tasks is higher than in casual tasks. The same result was observed in the fourth experiment. Finally, from historical data analyses, we found that Day Frequency and some usage logs are significantly different according to the page types.

Summary. We also briefly summarized all of the observed patterns as the following.

1) Generally, the amount of usage logs is not under the influence of the size and form of the Web page.

2) Information scents exert noticeable influence on usage patterns such that Web users choose links to visit based on information scents, and the scents also cause the users to show some uncertain usage patterns while they are viewing Web pages.

3) The viewing time is the best log to be used as an implicit feedback indicator if it is pre-processed carefully. It means that we have to analyze the viewing time more carefully than other logs to develop personalization services that are adaptive to user interest.

4) The viewing time is under the influence of interest and credibility levels. In other words, interest and credibility levels are the most influential contextual factors in a natural Web environment. The difficulty and complexity levels do not create noticeable variations on the amount of usage logs.

5) The viewing time is also under the influences of current tasks, written languages, and page types. In addition, page types are also influential on the variations of other usage logs such that the amount of mouse wheel use, number of visits in a day, and the amount of keyboard typing were significantly different based on the page types.

6) Web users visit different Websites when they are performing different tasks and they show different navigational patterns according to the task types.

7) We recognized that some historical and experiential aspects that may not be observed in short time analysis can only be found in long time analysis.

Limitations of the experiments

Although many interesting patterns were observed from our experiments, we also acknowledge the limitations of our experiments. We cannot expect that the observed patterns will generalize to a general population because we recruited small number of people from same population for our subjects according to our experimental convenience. However, the results show us valuable usage patterns of experienced Web users and consequently provide us with a good insight into further researches.

Future Work

As we already discussed in previous sections, the viewing time is under the influence of various factors. We cannot decide what service applications are to be activated based solely on the fact that viewing time increases on a current Web page, because the viewing time will be affected by various factors - interest levels, credibility levels, page types, tasks, and written languages. Therefore, to find a user's characteristics and select the applications accordingly, it is necessary to intelligently detect what factors are currently influencing the usage patterns. We think that it will be very challenging to find current contextual factors intelligently, but we also think that the current factors can be identified through some careful statistical analyses on various historical usage patterns. For example, as we already discussed in section 4.5, the URLs of the Web pages that users are currently viewing will give us information of the current task types. In addition, because Web users have a tendency to choose Websites to visit according to their own previous experiences about the sites, the URLs are also useful for inferring the users' subjective feedback levels on the contents of Web pages if we monitor user activities for a long period. Actually, in the post interviews of the third experiment, the subjects told us that they use different search engines according to their current tasks. For examples, they use Google for careful tasks and Naver – a Korean portal site - for casual tasks. Therefore, we assume that URL information can be used very effectively for the purpose of inferring the user's contexts. The similarity between the contents of current Web pages and contents of previous high-interested Web pages can also be used to infer the interest levels on the current Web pages. Furthermore, the Day Frequency can be used to infer the types of Web pages viewed.

If our system can infer the current contextual factors intelligently, some proactive services can be provided. In figure 10, we present the concept of a data preparation service that we are developing in which unnecessary visit logs and uninterested contents can be filtered out. In addition, if the system can identify a user's current task type correctly, the threshold of the viewing time to find high-interested Web pages can be applied accordingly.

Finally, we should consider individual differences because there may be variances according to user preference, cognitive styles, temperament, and so on.

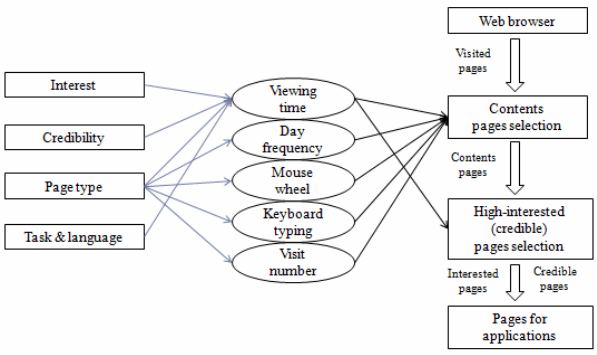


Figure 10. A possible practical solution- the arrows on the left shows their influential relationships and the arrows on the right means that the logs can be used for data preparation tasks

References

Al halabi W. S.; Kubat. M.; and Tapia M. 2007. Time spent on a web page is sufficient to infer a user's interest. In *Proceedings of the IASTED European Conference: internet and multimedia systems and applications*, 41-46, Chamonix, France: ACTA Press.

Badi R.; Bae S.; J. Moore M.; Meintanis K.; Zacchi A; Hsieh H.; Shipman F.; and Marshall C. C. 2006. Recognizing user interest and document value from reading and organizing activities in document triage. In *Proceedings of the 11th international conference on Intelligent user interfaces*, 218-225, Sydney, Australia: ACM.

Borlund, P. and Ingwersen, P. 1997. The Development of a Method for the Evaluation of Interactive Information Retrieval Systems. *Journal of Documentation*, 53(3):225-250.

Byström K. and Järvelin K. 1995. Task complexity affects information seeking and use. *Information Processing and Management*, 31(2):191-213.

Chi E. H.; Pirolli P.; Chen K.; and Pitkow J. 2001. Using information scent to model user information needs and actions and the Web. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, 490-497, Seattle, Washington, United States: ACM.

Choi J.; Lee G.; and Um Y. 2007. Analysis of Internet Users' Interests Based on Windows GUI Messages. In *Proceedings of the 12th International Conference on Human-Computer Interaction, Lecture Notes in Computer Science*, 4553:881-888.: Springer Berlin / Heidelberg.

Cooley R.; Mobasher B.; and Srivastava J. 1999. Data Preparation for Mining World Wide Web Browsing Patterns. *Knowledge and Information Systems*, 1(1):5-32.

Domenech J. M. and Lorenzo J. 2007. A Tool for Web Usage Mining. In Proceedings *of the 8th International Conference on Intelligent Data Engineering and*

Automated Learning, Lecture Notes in Computer Science, 4881:695-704, D.: Springer Berlin / Heidelberg.

Fogg B. J.; Soohoo C.; Da-nielson D. R.; Marable L.; Stanford J.; and Tauber E. R. 2003. How do users evaluate the credibility of Web sites?: a study with over 2,500 participants. In *Proceedings of the 2003 conference on Designing for user experiences*, 1-15, San Francisco, California: ACM.

Fu Y.; Shih M.; Creado M.; and Ju C.. Reorganizing web sites based on user access patterns. 2001. In *Proceedings of the tenth international conference on Information and knowledge management*, 583-585, Atlanta, Georgia, USA,: ACM.

Gauch S.; Speretta M.; Chandramouli A.; and Micarelli A. 2007. User Profiles for Personalized Information Access. The Adaptive Web, *Lecture Notes in Computer Science*, 4321:54-89,: Springer Berlin / Heidelberg.

Hofgesang P. I. 2006. Relevance of Time Spent on Web Pages. In *Proceedings of KDD Workshop on Web Mining and Web Usage Analysis, in conjunction with the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Philadelphia, PA.

Johnson J. D. 2003. On contexts of information seeking. *Information Processing & Management*, 39(5):735-760: Elsevier

Kari J. and Savolainen R. 2007. Relationships between information seeking and context: A qualitative study of Internet searching and the goals of personal development. *Library & Information Science Research*, 29(1):47-69: Elsevier

Kellar M.; Watters C.; Duffy J.; and Shepherd M. 2005. Effect of Task on Time Spent Reading as an Implicit Measure of Interest. In *Proceedings of the American Society for Information Science and Technology*, 41(1):168-175.

Kellar M.; Watters C.; and Shepherd M. 2007. A Field Study Characterizing Web-based Information Seeking Tasks. *Journal of the American Society for Information Science and Technology*, 58(7):999-1018: John Wiley & Sons.

Kelly D. and Belkin N. J. 2004. Display time as implicit feedback: understanding task effects. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, 377-384, Sheffield, United Kingdom: ACM.

Kelly D. and Cool C. 2002. The effects of topic familiarity on information search behavior. In *Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, 74-75, Portland, Oregon, USA.

Kelly D. and Teevan J. 2003. Implicit feedback for inferring user preference: a biblio-graphy. *ACM SIGIR Forum* 37(2):18-28. ACM.

Kelly D. 2004. Understanding implicit feedback and document preference: a naturalistic user study. Ph.D. Dissertation, Rutgers University. 2004.

Kim K. and Allen B. 2002. Cognitive and task influences on Web searching behavior. *Journal of the American Society for Information Science and Technology*, 53(2):109-119: John Wiley & Sons.

Liu Z. 2005. Reading behavior in the digital environment. *Journal of Documentation*, 61(6):700-712. Emerald Group Publishing Limited.

Mobasher B.; Cooley R.; and Srivastava J. 2000. Automatic personalization based on Web usage mining. *Communications of the ACM*, 43(8):142-151: ACM.

Nakamichi N.; Shima K.; Sakai M.; and Matsumoto K. 2006. Detecting low usability web pages using quantitative data of users' behavior. In *Proceedings of the 28th international conference on Software engineering*, 569-576, Shanghai, China: ACM.

Rieh S. Y. 2002. Judgement of information quality and cognitive authority in the Web. *Journal of the American Society for Information Science and Technology*, 53(2):145-161: John Wiley & Sons.

Salton G. and McGill M. J. 1986. Introduction to Modern Information Retrieval: McGraw-Hill.

Seo Y. W. and Zhang B. T. 2000. Learning user's preferences by analyzing Web-browsing behaviors. In *Proceedings of the fourth international conference on Autonomous agents*, 381-387, Barcelona, Spain: ACM.

Sonnenwald D. H. 1999. Evolving Perspectives of Human Information Behavior: Contexts, Situations, Social Networks and Information Horizons. Exploring the contexts of information behaviour, 176-190: Taylor Graham Publishing.

Vakkari P. 1999. Task complexity, problem structure and information actions: integrating studies on information seeking and retrieval. *Information processing & management*, 35(6):819-837: Elsevier.

Vakkari P. 2001. A theory of the task-based information retrieval process: a summary and generalisation of a longitudinal study. *Journal of Documentation*, 57(1):44-60: Emerald Group Publishing Limited.

Wang Y. D. and Emurian H. H. 2005. An overview of online trust: Concepts, elements, and implications. *Computers in Human Behavior*, 21(1):105-125: Elsevier.

Wang P.; Hawk W. B.; and Tenopir C. 2000. Users' interaction with World Wide Web resources: an exploratory study using a holistic approach. *Information processing & management*, 36(2):229-251: Elsevier.

Wathen C. N. and Burkell J. 2001. Believe it or not: Factors influencing credibility on the Web. *Journal of the American Society for Information Science and Technology*, 53(2):134-144: John Wiley & Sons.

White R. W. and Drucker S. M. 2007. Investigating behavioral variability in web search. In *Proceedings of the 16th international conference on World Wide Web*, 21-30, Banff, Alberta, Canada: ACM.

Exploiting Semantic Web Technologies for Recommender Systems A Multi View Recommendation Engine

Houda OUFAIDA, Omar NOUALI

DTISI Laboratory, CERIST Research Center
03, Rue frères Aissou - Ben Aknoun – Algiers, Algeria
{houfaida, onouali}@mail.cerist.dz

Abstract

Collaborative filtering systems are probably the most known recommendation techniques in the recommender systems field. They have been deployed in many commercial and academic applications. However, these systems still have some limitations such as cold start and sparsty problems. Recently, exploiting semantic web technologies such as social recommendations and semantic resources have been investigated. We propose a multi view recommendation engine integrating, in addition of the collaborative recommendations, social and semantic recommendations. Three different hybridization strategies to combine different types of recommendations are also proposed. Finally, an empirical study was conducted to verify our proposition.

Introduction

Dealing with information overload is one of the most challenging problems in the information access field; the Web is a perfect example. Unlike retrieval systems (Google, AltaVista, Yahoo, ….) which succeed in selecting suitable items according to a specific user query, these items are the same for every user in every situation, recommender systems aim to make personalized recommendation to users according to their preferences, tastes and interests expressed by users themselves or learned by the recommender system over the time.

There has been much work in this research area, from the early 1990 and still remains up to now. Foltz and Dumais experiences (Foltz and Dumais 1992) on four recommendation techniques have shown ambitious results, Resnick and collaborators proposed one of the first and probably the most known recommender system in the literature; Grouplens (Resnick et al. 1994) which recommends films to users according to their previous ratings.

Since, several models were proposed in the literature and much more applications were developed in the industry. Examples of such applications include e-commerce websites like Amazon.com for recommending books, CDs and different other items. MovieLens and Netflix for recommending movies and DVDs…

Recently, a new generation called semantic and social recommender systems have emerged taking advantage of the advancements in the semantic web technologies and features such as ontologies, taxonomies, social networks, tagging.

In this paper, we introduce a multi view recommender system that includes collaborative, social and semantic views of the user's profile. Each view recommends a set of items. Hence, three hybridization strategies are proposed for recommendations re-ranking. Finally, results from our experimentations are presented.

The rest of the paper is organized as follows: First we present the introduction of new Web 2.0 aspects in recommender systems. Then we expose our multi view recommender system, we present user's multi view representation and then present three recommendation modules: collaborative, social and semantic matching, hybridization strategies are also exposed. Finally, we discuss our experimental results and conclude with a summary of conclusions and outlooks.

Related Work

The key for an efficient recommender system is better understanding of both users and items. However, traditional recommender systems consider limited data (ratings, keywords) to compute predictions and do not take into account different factors necessary to understand reasons behind a user's judgment; is it the item's content, quality, is it because a friend recommended it?… Consequently, the users' classic communities' reflects only a *global* similarity usually insufficient to describe relations connecting users and even more items.

With the emergence of the Web 2.0, advancements allowed the apparition of a new generation of recommender systems: semantic and social recommender systems.

The availability of large product taxonomies on the Web (UNSPSC, Amazon.com, ODP for example) has encouraged the use of a taxonomy based user's/item's description in recommender systems. Quickstep (Middleton, Shadbolt, and De Roure 2004) used a paper

topic ontology, AKT-ontology, to extract weighted ontology topics as user's profile. (Lops, Degemmis, and Semeraro 2007) implemented k-means clustering algorithm for neighborhood generation based on semantic similarities between users. Each user's profile contains two semantic vectors; positive and negative weighted concepts extracted from Wordnet lexical database.

Mobasher and collaborators (Mobasher, Jin and Zhou 2004) propose an enhanced similarity measure which combine two measures; a semantic items' similarity and the classical rating similarity in a linear combination to perform recommendations. Moreover, (Wang and Kong 2007) calculate three similarity measures: collaborative, semantic and *demographic* similarities. An *offline* clustering algorithm is applied to reduce computation complexity.

Another promising aspect of the semantic Web is the items' tagging (Flickr, del.icio.us). Karen and collaborators (Karen, Marinho, and Schmidt-Thieme 2008) proposed to extend User × Item rating matrix with user tags as items and item tags as users. Szomszor and al. (Szomszor et al. 2007) proposed the use of collaborative tagging, also known as *folksomies*, to enrich users' profiles. Thus, each user has a tag cloud, as well as items. User's predicted interest on each tagged item can be made based on the semantic similarity between items' tags and user's tag-clouds.

The huge popularity of online social communities, such as Facebook (175 million registration), MySpace (110 million registration) has encouraged the use of user's social and personal data in recommendation process, especially in taste related domains (movies, music,).

The first idea about the way to introduce social networks in recommender system was to replace the similarity based neighborhood formation by social neighborhood (friends and friends of friends). (Sinha and Swiringen 2001) compared collaborative recommendations made by user's friends and those predicted by the system. The results showed that users prefer friends' recommendations. This can be explained by the fact that users *trust* their friends' choices.

(Groh and Ehmig 2007) conducted an empirical study to compare collaborative and social recommendations. The experiments have shown that social recommenders perform as good as the best collaborative filtering systems when data is sparse. Similarly, (Golbeck and Ziegler 2006) developed a social network website, FilmTrust, where users manage their FOAF (Friend Of A Friend Vocabulary) based profiles and used TidalTrust algorithm (Golbeck 2005) to infer trust values over the social network. The experimental results have shown that there is a strong correlation between trust relationships and profile similarities.

(Massa and Avessani 2004) presented a trust-aware recommender system named «Web of Trust» where users define a number of users they trust. This model uses the User × Item rating matrix and the User × User trust matrix and produces as an output a predicted User × Item rating matrix less sparse from the original one. Such method is particularly beneficial in new user recommendations

Proposed Approach

Seeking on greater understanding of user's choices and judgments, we propose a novel approach which introduces social and semantic levels into the recommendation process beyond the collaborative level. Hence combining collaborative recommendations with social and semantic ones is the key idea of our proposal.

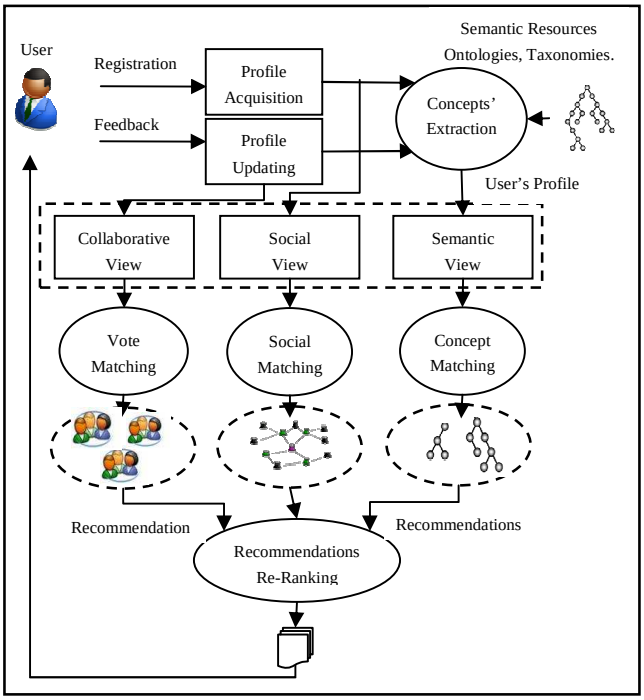


Figure 1: Multi view recommendation engine

User's Representation

Among the user's needs, user's profile is represented by three dimensions or views.

The *collaborative view* contains user's explicit or implicit ratings.

The *socio-demographic view* contains user's social data like age, gender, profession, location, personal and professional home pages, and friends' contact lists.

The *semantic view* represents user's interests in terms of a weighted concepts vector based on a hierarchical items' classification

Neighborhoods Generation

Each of the three views, proposed above, will be used by a recommendation engine to affiliate the user into a specific neighborhood and thus generate recommendations.

Collaborative Neighborhood. The collaborative view contains user's explicit or implicit ratings. Pearson Correlation can be used to compute users' similarities and k nearest neighbors' algorithm to determine such neighborhood in a classic way.

Social Neighborhood. Social recommendations are based on user's social community. It contains user's friends with trust values expressing how much the active user trusts his friends. The user annotates his relationships with such information. Trust can be binary (trust or don't trust) or on some scale, 1-5 scale where 1 is low trust and 5 is high trust. Based on these trust values, user's social neighborhood can be inferred over the social network. For example, Tidal Trust algorithm can be used (Golbeck 2006).

Semantic Neighborhood. Semantic view represents user's interests about items' content. For this, items' semantic content representation is needed.

Our choice was pointed on the use of a hierarchic semantic items' classification combined with user's evaluations to generate such view. The motivation behind this choice is the availability of such meta-information, like those of internet and e-commerce portals (Yahoo, Open Directory, LookSmart, Amazon, etc), where items are gathered into topics, which are themselves organized into a hierarchy going from the most general to the most specific.

We assume the existence of such classification H, where every item d is represented by a weighted concept vector C_d :

$$C_d = \left\{(c_1^d, w_1^d), (c_2^d, w_2^d), \ldots, (c_n^d, w_n^d)\right\}$$

The semantic view is a key element in our proposal; it is represented by weighted concepts vector C_u. These concepts are extracted from items' description C_d which the user has already rated.

$$C_u = \left\{(c_1^u, w_1^u), (c_2^u, w_2^u), \ldots, (c_m^u, w_m^u)\right\}$$

Concept's weight represents its interest score for the user. We propose the use of the weighted average to compute the concept's average rating expressing how much the user is interested in this concept; the result is divided by the maximum rating value Max_v (5 for example) to have a value between [0,1]

$$w(c) = \frac{\sum_j w_j r_{u,j}}{\sum_j w_j} \Big/ Max_r$$

User's vector C_u is updated when the active user rates a new item d. Hence, for each concept c contained in the new item's vector, there are four possible situations:

1. c already exists in C_u;
2. c is a super class concept of a concept in C_u;
3. c is a sub class concept of a concept in C_u;
4. c is a new concept, and is neither a super class nor a sub class concept of a concept in C_u;

We propose the following algorithm (Algorithm 1.) for semantic user's profile updating. It is executed for each new rating r:

Algorithm1: Profile Updating

Begin

Input $C_d = \left\{(c_1^d, w_1^d), (c_2^d, w_2^d), \ldots, (c_n^d, w_n^d)\right\}$ /* item's *d vector* */

$C_u = \left\{(c_1^u, w_1^u), (c_2^u, w_2^u), \ldots, (c_m^u, w_m^u)\right\}$ /* User's *u* vector */

$v_{u,d} = r$ /* user *u* rating on item *d* */

Foreach $c_i^d \in C_d \,|\, w_i^d \geq \min_{wd}$ **Do**

Switch c_i^d :

$c_i^d \in C_u$: /* c_i^d already exists in C_u */

$$w_{ui} = \frac{\sum_j w_j v_{uj} + w_i^d * r}{\sum_j w_j + w_i^d} \Big/ Max_v$$ /* c_i^d weight's updating*/

$\exists c_j^u \in C_u \,|\, c_i^d \in S(c_j^u)$ /* c_i^d super class concept of a concept in C_u*/

$C = \left\{c' \,|\, c' \in C_u \;\&\; c_j^u \in S(c')\right\}$

Foreach $c' \in C$ Do

$$w_{c'} = \frac{\sum_j w_j v_{uj} + w_i^d * sim(c', c_i^d) * r}{\sum_j w_j + w_i^d * sim(c', c_i^d)} \Big/ Max_v$$

End

$\exists c' \in C_u \,|\, c' \in S(c_i^d)$: /* c_i^d a sub class concept of a concept *c'* in C_u*/

$$w_{c'} = \frac{\sum_j w_j v_{uj} + w_i^d * sim(c', c_i^d) * r}{\sum_j w_j + w_i^d * sim(c', c_i^d)} \Big/ Max_v$$

$C_u = C_u \cup \left\{(c_i^d, \frac{w_i^d * r}{Max_v})\right\}$ /* adding c_i^d to C_u */

Else : /* c_i^d is a new concept */

$C_u = C_u \cup \left\{(c_i^d, \frac{w_i^d * r}{Max_v})\right\}$ /* adding c_i^d to C_u */

End

End

End.

In order to generate recommendations based on semantic view of the user's profile, users with similar interests must be found to build semantic neighborhood.

Hierarchical concepts organization allows us to reach users with similar concepts and those having more specific concepts in their semantic views. For example, in a hierarchic film classification, if we know that a user *u* likes "*comedy*" films in general, he should have concept "*comedy*" with a high interest weight, "*0.9*" for example, in his semantic view and there are other users which like more specific comedy kind films such as "*dark comedy*" or "*fantasy comedy*", these users should belong to the active user's neighborhood with a certain *membership degree*. (Algorithm 2.) builds such neighborhood ;

Algorithm2 : User Concept Matching

Begin

Input $C_u = \{(c_1^u, w_1^u), (c_2^u, w_2^u), ..., (c_m^u, w_m^u)\}$ /* User's u vector */

Foreach $c_i^u \in C_u \,|\, w_i^u \geq \min_{wu}$ **Do**

$V_{init} = \{u_j \,|\, c_i^u \in C_{uj}\} \cup \{u_j \,|\, c_s \in C_{uj} \;\&\; c_s \in subconcept\ s(c_i^u)\}$

Foreach $u_j \in V_{init}$ Do

$$\deg ree(u_j) = \frac{sim(c_i^u, c_{uj})}{|w_i^u - w_{uj}| + 1}$$

Priority_List_c $_i^u$ *.add(*u_j*,degree(*u_j*))*

End

$V_u = \{\cup_{i=1..m} \Pr ipority\_List\_c_i^u\}$

End

End.

The membership degree formula is proportional to the similarity between the two users' concepts and inversely proportional to the difference between their interest scores.

Thus for each concept with a significant weight (>=min_{wu}), we look for users having the same concept in their semantic views (*Vinit*) and users with more specific concepts, *Subconcepts(c)* function looks for such users (Algorithm 3.).

Algorithm3 : SubConcepts (c)

Begin

If (*depth(c)=depth(H)*) **then** /**c* is a leaf concept*/

$subconcepts(c) = \phi$

Else

If (*depth(c)=depth(H)-1*) **then** /**c* is a super class concept of a leaf concept*/

$subconcepts(c) = \{c' \,|\, c' \text{ IS-A } c \;\&\; \exists u \,|\, c' \in C_u\}$

Else

$subconcepts(c) = \{c' \,|\, c' \text{ IS-A } c \;\&\; \exists u \,|\, c' \in C_u\}$

$C = \{c'' \,|\, c'' \text{ IS-A } c' \;\&\; c' \text{ IS-A } c\}$

While ($subconcepts(c) = \phi$) **Do**

$subconcept\ s(c) = \{\cup_{c \in C} subconcept\ s(c')\}$

$C = \{c'' \,|\, c'' \text{ IS-A } c' \;\&\; c' \in C\}$

End

End

End

End.

Once semantic neighborhood built, remains rating predictions on items (Algorithm 4.).

Recommendations' Re-Ranking

Since each collaborative, social and semantic recommendation engines produce their own list of recommendations, recommendations' re-raking is required. The question here is "which hybridization strategy to adopt?" Burke (Burke 2005) experimented five hybridization strategies: weighted, switching, cascade, feature combination and feature augmentation hybrids. In this paper, we propose three possible hybridization strategies: *mixed, weighted* and *switched*.

Algorithm4 : Prediction

Begin

Foreach $c_i^u \in C_u$ **Do**

While *Priority_List_c* $_i^u$ .count > 0 **Do**

$$p_{u,j} = k \sum_{i=1}^{n} sim(u, u_i) v_{i,j}$$

with $$k = \frac{1}{\sum_{i=1}^{n} sim(u, u_i)}$$

and $$sim(u, u_i) = \frac{1}{\sum_{i<=m} w_i^u} \sum_{i<=m} w_i^u \deg ree_i(u, u_i)$$

End

End

End.

For this we introduce a confidence value per concept and per recommendation engine. This value represents how much a user likes items from a specific recommendation engine which are classified under this concept. The intuition behind this proposition is that a specific user *u* may like friends' recommendation for "*comedy*" films and semantic recommendations for "*documentary*" films for example.

Hence, for each concept in semantic view, we introduce three confidence values denoted as: F_{coll}, F_{soc} and F_{sem} for collaborative, social and semantic concept confidence. We compute the percentage of returned items that are relevant for each recommendation engine classified under a concept *c*:

$$F = \frac{\left\| \{ d / r_{u,d} >= R, c \in C_d, w_c >= W \} \right\|}{\left\| \{ d / c \in C_d, w_c >= W \} \right\|}$$

R is the minimum user's rating to be considered as relevant, 4 for example, and *W* is the minimum concept's weight in item *d* to be considered as significant, 0.7 for example.

For each concept in the semantic view, the three confidence values are maintained. Thus, the concept vector C_u is completed as follows:

$$C_u = \{(c_1^u, w_1^u, p_{coll1}, p_{soc1}, p_{sem1}), ..., (c_m^u, w_m^u, p_{collm}, p_{socm}, p_{semm})\}$$

For new concepts, the three confidence values are initialized as $F_{coll} = F_{soc} = F_{sem} = 1/3$.

Mixed Hybridization. Perhaps, the first idea that comes to mind is to simply mix recommendations from the three recommendation engines. If an item is recommended from more than one engine, the final rating is calculated as the average between each engine's rating. The following linear combination computes such average:

$$r_{u,d} = \alpha . r_{coll} + \beta . r_{soc} + \delta . r_{sem}$$

With: $\alpha = \beta = \delta = 1/n$ if d is recommended by *n* recommendation engines (n<=3). If a recommendation engine doesn't recommend *d*, its corresponding rating *r* will be 0.

Weighted Hybridization. Unlike the first hybridization strategy, α, β and δ values are proportional to the confidence values of recommended item's concepts. Hence, α parameter is computed as the weighted average of item's collaborative confidence values, as well as β and

δ. We propose the following algorithm to be applied to each resulting item (Algorithm 5.).

Algorithm5: Weighted Hybridization

Begin

Input $C_d = \{(c_1^d, w_1^d), (c_2^d, w_2^d), \ldots, (c_n^d, w_n^d)\}$ /* item's *d* vector*/

$C_u = \{(c_1^u, w_1^u, p_{coll1}, p_{soc1}, p_{sem1}), \ldots, (c_m^u, w_m^u, p_{collm}, p_{socm}, p_{semm})\}$ /* user's *u* vector*/

$$\alpha = \frac{\sum_j w_j^d \, p_{collj}}{\sum_j w_j^d} \qquad \beta = \frac{\sum_j w_j^d \, p_{socj}}{\sum_j w_j^d} \qquad \delta = \frac{\sum_j w_j^d \, p_{semj}}{\sum_j w_j^d}$$

/* $p_{collj} = p_{socj} = p_{semj} = 1/3$ **if** $c_j^d \notin C_u$ */

/* Normalization*/

$$\alpha = \frac{\alpha}{\alpha+\beta+\delta} \; ; \; \beta = \frac{\beta}{\alpha+\beta+\delta} \; ; \; \delta = \frac{\delta}{\alpha+\beta+\delta}$$

End.

Switched Hybridization. In this strategy, if an item is recommended from more than one recommendation engine, we chose the rating provided by the engine corresponding to the maximum value of item's global confidence values α, β or δ .

Experimental Evaluation

In order to experiment our multi view recommender system, we use *BookCrossing* dataset[1]. This dataset contains 42643 implicit ratings provided by 10000 users on 21944 books, which gives an average of 4.26 rating per user. These ratings were collected from *All Consuming*[2] website where people can share their interests about books, movies, food and other items. However, user's friends' list is not available, only user's age and location are available.

Amazon uses a hierarchy of nodes, called *Browse Nodes*, to organize its items for sale. Each node represents a collection of items, such as "Harry Potter books", not the items themselves. Browse nodes are related in a hierarchical structure.

Hence, for all rated books in the dataset, we crawled the Amazon web service for 15 days to get each book's nodes, the result was 309205 nodes including 6176 distinct node which gives an average of 14 nodes per book.

However, Amazon does not provide nodes' weights, for this and in order to favor most specific nodes and at the same time to diminish the weight of nodes that occur very frequently, we have estimated node's *i* weight as follows:

$$Weight(i) = \frac{(\frac{depth_i}{Max_{depth}} * \log(\frac{N}{n_i}))}{Max_{weight}}$$

With $depth_i$ is node's *i* depth in Amazon's classification, Max_{depth} is the depth of the most specific node of the current item, *N* is number of items classified under the root node "*books*", n_i is number of items classified under node *i* and finally, Max_{depth} is used to normalize all resulting weights values for the current item. We also used *Lin* semantic similarity for this evaluation.

Our evaluation methodology was as follows. User's collaborative, social and semantic views are built. Collaborative view contains user's ratings. Since, user's friends' list data is not available; we have simulated such neighborhood by considering users living in the same location and having similar ages. For the semantic views, we have generated different user's semantic views depending on ratings number considered; seven collaborative and semantic views are constructed for each user for 1, 5, 10, 20, 30, 40, 50 ratings considered. The social view remains the same since it does not depend on user's ratings.

We have varied the number of ratings considered for the recommendation generation and then measured recommendation accuracy using MAE measure and coverage using RECALL measure, applied on each recommendation engine separately and also with mixed hybridization strategy .

For each recommendation list, we have calculated the average of MAE and Recall values for Top5, Top10, Top20, Top30, Top40 and Top50 items. Figure 2 displays our results.

Preliminary results show that in term of precision, semantic recommendation engine produce more accurate recommendations comparing it to collaborative engine, especially with small nucmber of ratings (<10) however in terms of recall, collaborative engine recommends more relevant items. Semantic engine bad recall may in part be explained by the fact that *SubConcept* function was limited at one level, i.e. we have only considered direct subclasses in user's neighborhood generation.

Mixed hybridization strategy appears to compromise between semantic recommendations good precision and collaborative recommendations good recall. It outperforms collaborative engine in terms of recall and keeps in the same time a good accuracy comparable to the semantic recommendation engine (Figure 3.).

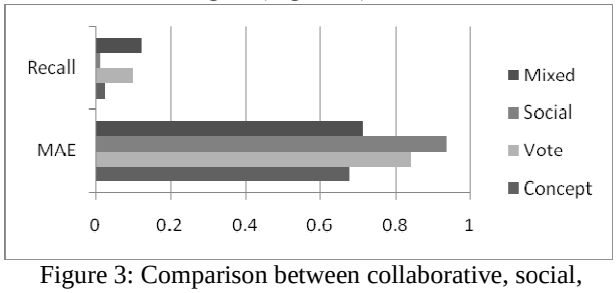
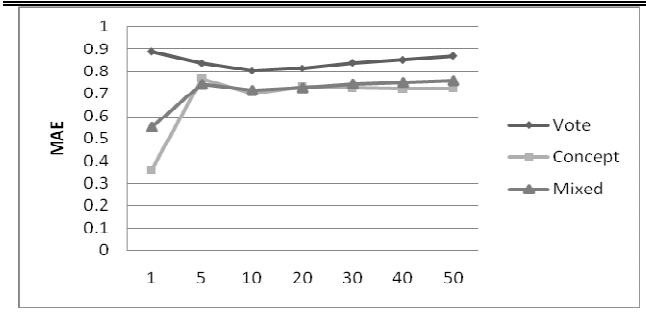


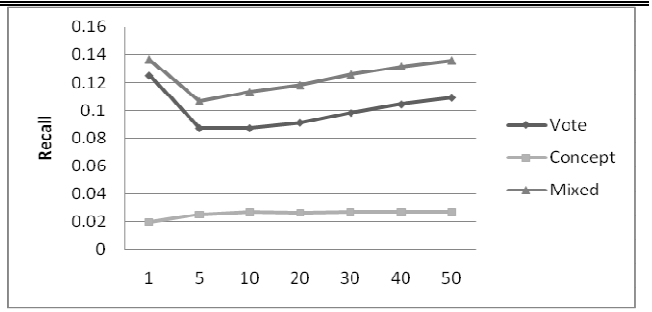
Figure 3: Comparison between collaborative, social, semantic and mixed recommendation engines

[1] http://www.informatik.uni-freiburg.de/%18cziegler

[2] http://www.allconsuming.net/

Figure 2: TopN MAE and Recall for collaborative, semantic and multi view recommendation engines





Conclusion

In this paper, we have proposed a multi view recommendation engine which exploits semantic web technologies such as semantic items' description and social networks beyond the classic ratings data. The results of our experimentations were very promising and improved the recommendation process in many ways:

1. Exploiting semantic background knowledge enriches description of different system elements (users, items);
2. Enhanced semantic description improves items' classification and users' clustering, it helps the system to produce more accurate predictions;

We believe that the introduction of a semantic level in recommender systems explains users' judgments in a semantic way and should lead to a greater understanding of the target users.

Social elements are particularly benefit in taste related domains. Our multi view recommendation system could make semantic enhanced predictions for an item's category (scientific papers for example) and social enhanced recommendations for another item's category (music, movies) if the user prefers that. Thus, experimenting this proposition in an online study will be interesting; it constitutes one possible outlook to investigate.

The use of interesting Web services which provide social data about users based on unified user's models (FOAF, APML for example) is also another interesting issue to investigate. Social communities may increase trust over recommender systems and encourage users to communicate with like-minded people. Thus, this consistent users' participation provides more information about their interests and preferences;

References

Burke, R. 2005. *Hybrid Systems for Personalized Recommendations.* Book chapter: Intelligent Techniques for Web Personalization.133-152, Springer.

Foltz, P.W. , and Dumais S.T.1992. *Personalized Information Delivery : An Analysis of Information Filtering Methods*. Communications of the ACM 35 (12), 51-60.

Golbeck, J. 2005. *Computing and Applying Trust in Web-based Social Networks*. Ph.D. thesis. University of Maryland. College Park, MD USA..

Golbeck, J., Ziegler, C.N. 2006. *Generating Predictive Movie Recommendations from Trust in Social Networks*, In Proc. of the fourth international conference on trust management.

Groh, G., and Ehmig, C. 2007. *Recommendations in taste related domains: Collaborative Filtering vs. Social Filtering*, ACM GROUP'07.

Karen, H.L.; Marinho, L.B.; and Schmidt-Thieme L. 2008. Tagaware *Recommender Systems by Fusion of Collaborative Filtering Algorithms*, ACM SAC'08.

Lops, P.; Degemmis M. ; and Semeraro, G. 2007. *Improving social filtering techniques through WordNet-Based user profiles*. UM 2007.

Massa, P., and Avesani P. 2004. *Trust-aware Collaborative Filtering for Recommender System*. In "On the Move to Meaningful Internet Systems: CoopIS, DOA, and ODBASE". Berlin, Heidelberg: Springer, pp. 3-17.

Middleton, S.E.; Shadbolt, N.R.; and De Roure, D.C. 2004. *Ontological User Profiling in Recommender Systems*. ACM Trans. Information Systems, vol. 22, no. 1, pp. 54-88.

Mobasher, B.; Jin, X.; and Zhou, Y. 2004. *Semantically enhanced collaborative filtering on the Web*. Book chapter. Web Mining: FromWeb to SemanticWeb.

Resnick, P.; Iacovou, N.; Suchak, M.; Bergstrom; P.; and Riedl, J. 1994. *GroupLens: An open architecture for collaborative filtering of netnews*. In Proc. of the 1994 Conference on Computer Supported Collaborative Work, Eds. ACM Press, New York. 175-186.

Sinha, R., and Swearingen, K. 2001. *Comparing recommendations made by online systems and friends*. DELOS-NSF Workshop on Personalization and Recommender Systems in Digital Libraries.

Szomszor, M.; Cattuto, C.; Alani, H.; O'Hara, K.; Baldassarri, A.; Loreto, V.; and Servedio, V.D.P. 2007. *Folksonomies, the Semantic Web, and Movie Recommendation*, In Proc. Of the ESWC'07.

Wang, R.Q., and Kong F.S. 2007. *Semantic-Enhanced Personalized Recommender System*. In Proc. of the international conference on machine learning and cybernetics.

Intelligent Web Navigation Using Virtual Assistants

Eduardo M. Eisman, Víctor López, Juan Luis Castro
Department of Computer Science and Artificial Intelligence
University of Granada (Spain)
{eisman,victor,castro}@decsai.ugr.es

Abstract

Some time ago, companies and organizations did not store very much information about themselves on the Internet. However, the Web has evolved a lot over the last years and websites contain more and more information. In many cases, that information is not well organized and users have to waste their time looking for what they want to know using the traditional menu-driven navigation and keyword search that websites provide. This decreases users' interest in surfing websites and hence in finding out about companies. One way to resolve this problem is to change the structure of websites. However, if a company has just spent a lot of money on redesigning its website or it simply likes its current design, it will refuse to change the way contents are organized. For this reason, we propose a real time Virtual Assistant, which can work together with any existing website, to help users find the information that they look for. Preliminary experiments have shown that Virtual Assistants can outperform traditional navigation.

1 Introduction

With the arrival of the digital era and the development of the Internet over the last years, there are more and more companies that store all the information they own in a digital medium and publish it on the Internet, so that everybody can have access to it. However, due to the big amount of information, it is very difficult for them to structure their contents in a logical way so that they can be accessed using only one or two mouse clicks. Nowadays, this objective is usually a utopia because users have to waste their time looking for what they want to know using the traditional menu-driven navigation and keyword search that websites provide. If the information to handle is small, this problem can be resolved changing the structure of the contents. However, as the information grows, it is more and more difficult to provide users with an easy and fast access to all that data. The necessity of new ways of accessing that information becomes evident.

One of the reasons that make users spend a lot of time wandering through websites to get some useful information arises from the diverse nature of the Internet. Nowadays, there are millions and millions of websites about different topics, and each website is different from the others. For this reason, there is a learning curve when you visit a website for the first time, and the steepness of the slope depends on how well organized the contents are and how skillful the user is. People who are used to working with new technologies can find the information that they are looking for in a website very quickly, even though they have never visited it before. However, other users find it very difficult to navigate through a website using menus to discover new information, and they need somebody to support them.

All these problems make it evident the necessity of an efficient and effective mechanism for organizing and accessing the information of a company when it becomes very big. The new system must be real time. Users' time is highly valuable, so if they had to wait every time they wanted to get some information, the system would be useless. It must be effective, that is, it must achieve the goals and objectives that users want. In this sense, the quality of the results must be high. Moreover, the communication between the system and users must be natural, allowing complete natural language questions rather than simple keywords. Finally, the interface must keep as simple as possible so that everybody can use it, regardless of their computer skills. To sum up, the system must make the access to information easier for everybody.

The rest of this paper is organized as follows. We begin with presenting some related work. We then describe the main features and advantages of our Virtual Assistant. We explain the architecture and the different modules of our system in detail. Then, we describe the user interface and its functionality. Afterwards, we carry out a comparative analysis in order to evaluate the performance of our system in comparison to traditional navigation. Finally, we conclude the paper and provide some directions for future work.

2 Related Work

As far back as the mid-nineties, people became seriously concerned about the recent explosive growth of the Web. Lieberman [1995] pointed out the necessity for some sort of intelligent assistance to a user browsing for interesting information. He introduced a behavior-based interface agent, Letizia, which tracked the user's browsing behavior and attempted to anticipate items of interest. Thus, the system suggested links to other documents, determining an ordering of interest

among them and providing a reason for making those choices, which decayed over time. However, it did not have natural language understanding capabilities.

Wexelblat and Maes [1999] proposed a set of tools, called Footprints. They said that buying a book is not the same as borrowing it. It is the same book (same words, pictures, and organization), but the borrowed book has additional information (notes in the margin, highlights, underlines, dogeared pages, and so on) and reflects its history because it opens more easily to certain places. These traces should be accessible to future users who could take advantage of the work done in the past. With this aim, Footprints showed the traffic through a website using a graph in which nodes were documents and links were transitions between them.

Cassell *et al.* [2000] created a very complete example of Embodied Conversational Agent (ECA) called REA (Real Estate Agent) which played the role of a real estate salesperson. However, it needed a lot of sensors and computational resources so it was not portable.

Abbattista *et al.* [2004] presented SAMIR (Scenographic Agents Mimic Intelligent Reasoning), which consisted of a 3D face, a custom version of the ALICE (Artificial Linguistic Internet Computer Entity) chatterbot [http://www.alicebot.org/], and a classifier system to keep conversation and face expressions coherent with each other.

Kim *et al.* [2005] proposed an information retrieval assistant, CHATTIE, which used natural language and could be integrated with outer information provision systems such as conventional information retrieval and relational database management systems in order to fill some slots in the answer.

AIML (Artificial Intelligence Markup Language) is an XML dialect for describing conversational scenarios for ECAs [Wallace, 2004], which specifies pairs of patterns and templates, so the agent answers the template associated to the pattern that best matches the question. Traditionally, programmers use AIML to create conversational rules by hand, considering the contents of web pages. However, every time a page is updated, related conversational rules have to be modified, and this is a problem when many pages are frequently modified. In order to avoid this problem, Kimura and Kitamura [2006] used RDF (Resource Description Framework) to represent the semantic contents of web pages.

Pilato *et al.* [2008] proposed an intelligent tutoring system for the Java programming language. When a student asked a question, the system looked for the concept that best matched that question and then created a backward path from that concept to the student's current knowledge state. The concepts and their relations were stored in a hierarchical ontology. Relations could be strong (prerequisites) or weak (something related). Most of the subjects were mandatory, although some of them were optional. Each document had a list with the most frequent non-stopwords, used to cluster documents and evaluate if a student knew the concept referred by a document. The learning path was the list of all the unknown nodes.

In conclussion, many of the existing systems, especially those from the earlier years, lack a virtual character which can engage in conversation with users, making the interaction process friendlier. In addition, some systems do not allow users to ask natural language questions. They should also take into account the context during the dialog, so that users could omit the implicit words in the conversation. Moreover, some others systems focus on the navigation history, only trying to organize the pages that have been already visited, instead of recommending related web pages to continue the navigation. These and other problems make the navigation process much more difficult. For this reason, we propose a natural language Virtual Assistant which covers all those features.

3 The Virtual Assistant

The Virtual Assistant is an intelligent system for supporting users when they look for some information in a website. It is a real time system because we must not keep users waiting for getting the information they want. Users engage the system in conversation using natural language queries, as if it was a real assistant, so it is really easy for users without computer skills, and much better than the one offered by traditional websites, where users can only click on static menus and do searches specifying some keywords. In addition, the aim is not only to answer users' questions but also offer recommendations that lead users and keep the conversation going. Context is another important feature. It allows users to omit some words if they are implicit in the conversation.

Next, we explain the architecture of the system and each module in detail, and discuss the features of the user interface.

3.1 The Architecture

Making the access to information easier for any kind of user should be the main objective of our system. For this reason, the Virtual Assistant has been designed using a client-server architecture, as can be seen in Figure 1. In this way, all the computation is done on the server side, so users only need a web browser. This picture also gives us an overview of how the system generates an answer for a specific query. First of all, the user's query is taken from the client to the server using the AJAX (Asynchronous JavaScript And XML) technology. There, it is processed by the Natural Language Understander (NLU) in order to identify the Information Units (IUs) to which it refers (in the next section we explain the concept of Information Unit in detail). Then, the Dialog Manager (DM) receives a list with the identified IUs, including the matching degree between each unit and the query. The DM uses a filter to include some of the IUs from that list in the backpack, which is the structure employed to store all the IUs about which it can talk in the future. Afterwards, the DM chooses the most suitable IU for that moment taking into account the previous dialog, and updates the backpack with the recommendations associated to that IU. Once it has been decided about which IU the Virtual Assistant is going to talk, the DM decides what to say about it. So, it creates a new action for that IU. There are different types of actions: inform, inform and suggest, ask for clarification (if the query is ambiguous), and ignore. Next, the Communication Generator (CG) retrieves and adapts a specific answer which fulfills the selected action. Finally, the generated answer is sent back to the user at the client side using AJAX.

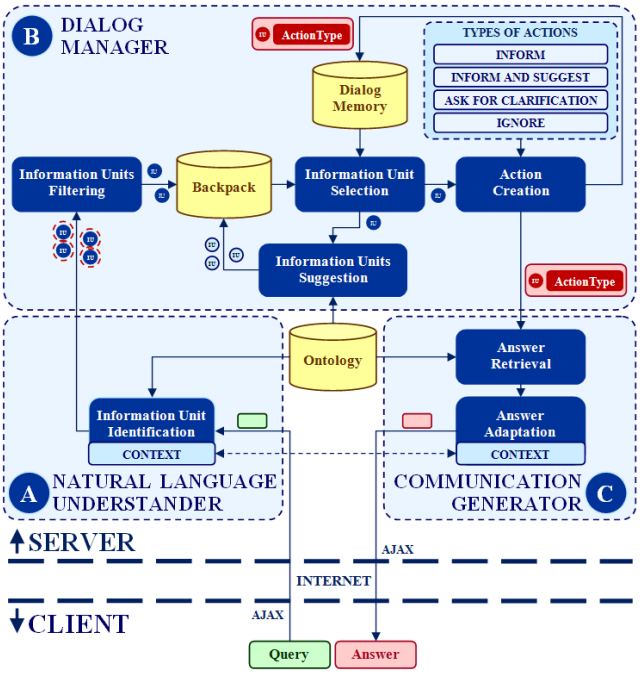


Figure 1: The Architecture

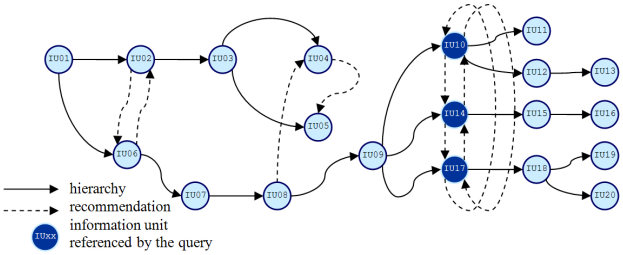


Figure 2: A simplified overview of the ontology

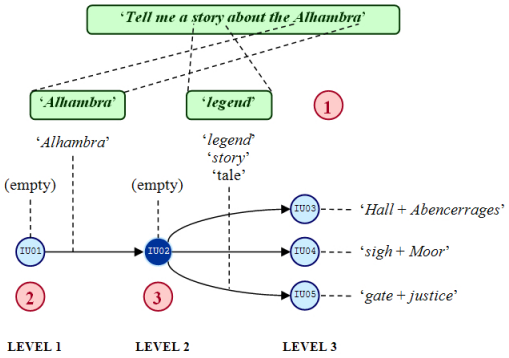


Figure 3: The IU identification process

3.2 The Ontology

All the knowledge about the domain is stored by the system using an ontology. Essentially, an ontology is a formal representation of a set of concepts within a particular domain and some relationships established between those concepts. Our ontology is based on entities called Information Units (IUs). An IU is a piece of information about a specific concept which has a meaning by itself. It includes the definition of the concept and also some meta information like its name, different ways that people can use to ask about it, a URL where more information about it can be found, and so forth. We can distinguish between two different types of IUs, objects and properties. The main different between them is that objects may have properties, but properties must not have either objects or properties. The reason for such a distinction is double. First, we can define templates for the objects of our domain, specifying the names of their properties, so that we do not have to define the properties every time we create a new IU of this kind of object. Second, as we distinguish between objects and properties, when a user asks a question we can identify both of them individually. Therefore, if the question is ambiguous and only the name of the property can be identified, the system can urge the user to concrete the object.

As we can see in Figure 2, IUs are connected with each other. Consequently, we can see our ontology as a kind of graph in which the IUs are the nodes and the connections between them are the edges of the graph. This special graph has some particular features. First, there is a hierarchy defined over it. Thus, IUs can be organized according to different criteria, such as objects included inside other objects, specialization, or any other kind of relationship between them (what we call recommendations).

3.3 The Natural Language Understander

When a user asks a question, the first thing that the system has to do is to identify the IUs that are referred by the question, in order to be able to generate an appropriate answer later on. This is the role of the Natural Language Understander (NLU). Its input is a natural language query, and its output is a list with the IUs that are referred or might be being referred by the query, and which are used by the Dialog Manager (DM) to determine about what to talk, when and how.

The first objective of the NLU is to recognize the keywords of the query that are going to be useful in the next phase. This process is not limited to the textual selection of keywords but it goes further, allowing the use of synonyms, derived words, and so on. The aim is to reduce the language to a specific vocabulary so that it can be easily managed by the system. Remember that our system must be real time. Figure 3 shows an example of IU identification process for a Virtual Assistant for the Alhambra monument. When the user asks the question '*Tell me a story about the Alhambra*', the NLU replaces the word '*story*' by '*legend*', and the word '*Alhambra*' remains constant. These two keywords are used to identify the properties and objects related to the query.

Each property has a set of keywords that represents different ways that people can use to talk about it. In this way, the system moves through all the property names, that are placed at the same level, checking whether they match the keywords or not (each matching word adds one vote to the property).

The process of generating the list of identified objects is different. Since there are much more objects than property names in the ontology, the system must avoid having to move

through all of them in order to be time and memory efficient. For example, if we consider the object '*Legends of the Alhambra*', a user could refer to it saying '*Tell me a tale about the Alhambra*', '*I would like to know a story about the Alhambra*', '*Do you know any legend about the Alhambra?*', and so on. Therefore, we could link the following keyword set to that object: '*tale + Alhambra*', '*story + Alhambra*', and '*legend + Alhambra*'. To avoid redundant information, the system makes good use of the hierarchical structure that objects have in the ontology. In this way, each object has a specific keyword set associated to it, and also an additional keyword set that all its descendants must match. In Figure 3, the specific keywords are linked to the IUs whereas the common keywords are linked to the connections between them. Hence, the system uses these two different sets to move through the hierarchy. If we go on with the example, the NLU starts from the first level, where both the specific keywords of the level (none in this case) and the keywords that all its descendants have in common ('*Alhambra*'), appear in the query. Therefore, the NLU marks this IU about the Alhambra as a candidate to be referred by the query, and descends to the second level through that branch. In this point, the two keywords sets of the IU also match the query, so the NLU adds a new candidate object and descends to the third level, where no keywords match. Consequently, at the end of the process, the NLU has identified two candidate objects which match the user's query ('*Alhambra*' and '*Legends of the Alhambra*').

Once the NLU has identified a list of property names and a list of candidate objects, it joins them, creating new IUs and putting them on the final list of identified IUs. If the NLU cannot identify any logical connection between them, it puts all that objects on the final list and passes the buck to the Dialog Manager (DM). The same applies when the NLU cannot identify any object but only some property names.

Context is another thing to take into account. If our system did not support it and the user wanted to know the price of the tickets for the museum, the question would be '*How much do the tickets for the museum of the Alhambra cost?*' instead of being '*How much do the tickets cost?*', which is much more natural. In order to resolve this problem, the NLU does not start the search from the top IU in the hierarchy but from the last IU that has been transmitted to the user. In this way, the NLU explores the graph below the last IU. After that, it moves to the parent IU and repeats the process but without exploring the last branch again. The searching process continues until the NLU reaches the top IU in the hierarchy.

3.4 The Dialog Manager

The Dialog Manager (DM) makes decisions about what the Virtual Assistant must do, when, and how. Its input is the list of IUs that match the user's query and its output is an abstract action which is later transformed into a specific answer by the Communication Generator (CG).

An action is made of an IU (the concept about which we want to talk) and an action type (the kind of information to provide about that concept). For example, we might want to talk about a particular concept, ask the user for clarification about a set of identified objects or properties, or deny answering about a concept because we have already talked about it. So, we can define different types of actions and specify the behavior of the Virtual Assistant for each one.

Two structures are used during this process. The memory, a chronological list with the actions that have been performed, lets us know if an action has been performed recently or not, how many times we have talked about an IU, and what we have said about it. The backpack stores the IUs about which we have planned to talk in the future, so we can lead the conversation instead of simply answering the user's queries. As we have already mentioned, the DM filters the IUs identified by the NLU so that only the most specific ones are included into the backpack. Next, in order to generate the new action, the DM chooses the next IU from the backpack. Afterwards, some IUs related to the selected one are included into the backpack in case we want to continue talking about related items in the future. Finally, the DM chooses what to say about that IU, that is, it selects the type of action to generate. This new action is added to the memory and sent to the Communication Generator, which generates a specific answer.

3.5 The Communication Generator

The Communication Generator (CG) transforms the abstract action provided by the Dialog Manager (DM) into a specific answer. In other words, it looks for the specific words that are going to be used to carry out that action. Whereas the action is language independent, the answer is language dependent, and it can be expressed in different languages. So, the Virtual Assistant is multilingual.

The answers of the system are templates that can contain up to three different information categories, depending on their updating frequency. Information can be static, immediate, or dynamic. Static information always appears in the answer. It can be fixed (e.g. the history of the Alhambra will always be the same, although each time the specific words of the answer might vary) or variable (e.g. the schedule of the Alhambra is different from November to March than from March to November, so, depending on when the user poses the query, the answer will be either the former or the latter). Immediate information may change quite often, so it should be stored in a database to make updating tasks easier (e.g. the price of the tickets for the Alhambra). Finally, dynamic information is automatically included in runtime (e.g. the time of the system at the moment of generating the answer).

3.6 The User Interface

There is a famous quotation from Albert Einstein which says '*make things as simple as possible, but no simpler*'. Following this line, Figure 4 displays a web page which can be divided into two different areas. Paying special attention to the top of the page, we can identify three areas. On the left side, we can see the Virtual Assistant. She can move the head, blink, and also speak. The text associated to the answer is transformed into speech using the Loquendo program [http://www.loquendo.com/]. This makes the conversation much more natural and similar to a real dialog. In the middle, there are two icons that allow the Virtual Assistant to lead the conversation instead of having to wait until the user asks any question. In addition, there is an input field for asking questions. Finally, on the right side, there are three main

Figure 4: The user interface

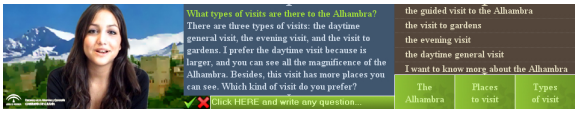


Figure 5: The user interface in action

topics about the application domain. Besides, once the user has asked a question, we can see in Figure 5 how the system modifies the interface in order to show the transcription of the generated answer as well as a list with some different topics related to it, on which the user might click. Finally, the bottom area is used to show web pages related to the query.

4 Comparative Analysis

In order to prove the reliability, effectiveness and efficiency of our system, we present an analysis that compares the Virtual Assistant to the traditional menu-driven navigation and keyword search that websites provide. In particular, we consider the website of the Alhambra (http://www.alhambra-patronato.es/) for this purpose. In testing a system which is designed to support users with an imprecise task such as browsing, it is difficult to find useful measures. We have considered three different tests, and we have employed fifteen IUs about the Alhambra that form a representative subset of all the IUs contained in the ontology of our system: the Alhambra (01), the Generalife (02), the history (03), the schedule (04), the telephone number (05), the types of visits (06), the places to visit (07), the Court of the Lions (08), the Palace of Charles V (09), the Hall of the Abencerrages (10), the prices of the tickets (11), the purchase of tickets (12), the accesses (13), the exhibitions (14), and the museum (15).

All these tests have been carried out on an Intel Pentium IV at 2.40 GHz, with 1.00 GB of RAM, Microsoft Windows XP, and Apache Tomcat.

Test One: The Menu-Driven Navigation

The first test compares the navigation using the recommendations proposed by the Virtual Assistant to the navigation using the menus of the website of the Alhambra.

Table 1 shows the minimum number of clicks that a user has to make in order to have access to the IU associated to the identifier of the corresponding column of the table. As we can see, the difference between both systems is very small. This is a menu-driven navigation and, in general, you can go wherever you want with two or three mouse clicks at the most. In the case of the telephone number of the Alhambra (IU05), the number of clicks is 0 because it appears at the bottom of all the web pages. Actually, through the *sitemap* of the website of the Alhambra, we can have access to any of those IUs with only two mouse clicks. However, users without an advanced knowledge about computers do not usually know what a sitemap is and therefore it is highly improbable that they will use this feature of the website. So, if we consider the averages that appear in Table 1, the Virtual Assistant performs slightly better (20%) than the website of the Alhambra. The main reason is that the website contains more information about the Alhambra than our system and for this cause it uses more levels to organize the data. However, this makes the usual information more difficult to be found and users might easily get lost along the way. In other words, two mouse clicks on the Virtual Assistant take less time than two clicks on the website of the Alhambra, because in the second case you have to find out to which category the information that you are looking for belongs.

The problem of identifying the category of the information that we are looking for deserves special attention. We are considering the best possible situation, that is, users know perfectly where all the information is placed in the website, even though it is the first time they visit it. However, this is an ideal situation because they often wander through websites to get some useful information, and this problem becomes more apparent as the user's computer skills decrease. For the users who have a high skill level (experts), the improvement achieved by the use of virtual assistants is really low because they are used to navigating with traditional menus. What is more, they could prefer traditional navigation because they can find the information faster. However, as the skill level decreases, users find it more difficult to move through websites, and here the Virtual Assistant can be really useful.

Using an analogy, the Virtual Assistant is like a Global Positioning System (GPS). When people are used to driving in a specific city, and they always go to the same places, they do not need a GPS. What is more, they could find it very stressful if they wanted to go to a place taking a short cut, and the GPS recommended another alternative path because it thought that it is the best. However, nobody can throw doubt on the fact that GPSs are really useful for people visiting a city which is completely unknown for them, because they can guide them through such an amount of streets.

Test Two: The Searching Process

The second test analyzes the searching power of both systems, considering the amount of time that searches take.

In order to measure the speed of the search engines, we have queried both systems about the fifteen IUs proposed. The results have shown that the Virtual Assistant outperforms in time the search engine of the website of the Alhambra in 340%. The main reason of such a strange huge difference is that the website probably performs a bad sequential search, reading all the documents once. On the contrary, our system

Table 1: Number of clicks needed to reach some IUs

System	# Information Unit (IU)															
	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	Average
Virtual Assistant	1	3	2	2	2	1	1	2	2	2	2	2	2	2	3	1.9
Alhambra's website	1	2	2	3	0	3	2	3	3	3	3	3	2	3	3	2.4

Table 2: Number of results per query found in the website

ID	Query[1]	#Docs[2]	#Docs[3]	%[4]
IU01	Alhambra	0	20853	-
IU02	Generalife	81620	8320	89.8
IU03	history	19768	1583	92.0
IU04	schedule	1118	395	64.7
IU05	telephone number	183	9	95.1
IU06	types visits	5	3	40.0
IU07	places	1036	9	99.1
IU08	Court Lions	20803	1063	94.9
IU09	Palace Charles V	58953	5169	91.2
IU10	Hall Abencerrages	2245	32	98.6
IU11	prices tickets	97	3	96.9
IU12	purchase tickets	1	1	0.0
IU13	accesses	24388	6464	73.5
IU14	exhibitions	13975	17	99.9
IU15	museum	105898	11252	89.4
	TOTAL	330090	55173	83.3

[1] Translation of the query from Spanish
[2] Results found in all the website
[3] Results found in the specific category to which the IU belongs
[4] Reduction in the number of results

takes advantage of the hierarchical structure of the ontology to index the search. However, these results must be handled carefully because the website stores more information than our system, so the comparison is not completely fair.

Another point that deserves special attention is that the Virtual Assistant allows contextual natural language questions rather than simple keywords, as it happens with the website.

Test Three: The Quality and Reachability of the Information

The aim of this test is to evaluate the quality of the information provided by both systems, that is, if it is relevant to the query and users do not have to waste their time looking for what they want to know among all that information.

With regard to the amount of information retrieved, Table 2 shows the number of results found by the website of the Alhambra for each query. Note that the results for the query '*Alhambra*' in the first search are really 0 (perhaps too many documents for that search string). In general, if we limit the search space, the reduction in the number of documents is over 80%. Even so, there are too many results. However, what is really important is not the amount of documents retrieved, but the fact that some times the results do not contain the requested information. In the case of the Virtual Assistant, the answer is always immediate.

5 Conclusion & Future Work

In this work we have studied the problem of the reachability of the information on websites. When the information grows, it is difficult to organize it in an appropriate way so that users can easily find it. For this reason, we have presented an intelligent natural language Virtual Assistant which makes the access to information easier specially for inexperienced users.

Regarding the future work, we must make the Virtual Assistant much more dynamic and with different personalities, so that it could behave in different ways and choose between several paths for generating the answer. Another point to take into account is the adaptation of the recommendations to the particular user of the system, according to his/her preferences. Finally, it would be a good idea to assist the navigation all over the website, not only when the user interacts with the system in a direct way.

Acknowledgments

This work has been supported by the Spanish Ministry of Science and Technology under Research Project TIN2007-67984-C02-01, the Andalusian Government under Research Project TIC-P06-01424, and a FPU scholarship from the Spanish Ministry of Science and Innovation.

References

[Abbattista *et al.*, 2004] F. Abbattista, G. Catucci, G. Semeraro, and F. Zambetta. Samir: A smart 3d assistant on the web. *PsychNology Journal*, 2(1):43–60, 2004.

[Cassell *et al.*, 2000] J. Cassell, J. Sullivan, S. Prevost, and E. Churchill. *Embodied conversational agents*. MIT Press, Cambridge, MA, USA, 2000.

[Kim *et al.*, 2005] H. Kim, C. N. Seon, and J. Seo. A dialogue-based information retrieval assistant using shallow nlp techniques in online sales domains. *IEICE - Trans. Inf. Syst.*, E88-D(5):801–808, 2005.

[Kimura and Kitamura, 2006] M. Kimura and Y. Kitamura. Embodied conversational agent based on semantic web. In *PRIMA*, pages 734–741, 2006.

[Lieberman, 1995] Henry Lieberman. Letizia: An agent that assists web browsing. In Chris S. Mellish, editor, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 924–929, Montreal, Quebec, Canada, 1995. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA.

[Pilato *et al.*, 2008] G. Pilato, R. Pirrone, and R. Rizzo. A kst-based system for student tutoring. *Applied Artificial Intelligence*, 22(4):283–308, 2008.

[Wallace, 2004] R. Wallace. The elements of aiml style. alice ai foundation, 2004.

[Wexelblat and Maes, 1999] A. Wexelblat and P. Maes. Footprints: history-rich tools for information foraging. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 270–277, New York, NY, USA, 1999. ACM.

Collaborative Filtering With Adaptive Information Sources

Neal Lathia
Department of Computer Science
University College London
Gower Street, WC1 E6BT, UK
n.lathia@cs.ucl.ac.uk

Xavier Amatriain, Josep M. Pujol
Telefonica Research
Via Augusta 177
Barcelona 081290, Spain
xar, jmps@tid.es

Abstract

Collaborative filtering (CF) algorithms, which generate recommendations for web users by predicting user-item ratings, are often evaluated according to their predictions; in this context the problem of generating recommendations can be formulated as one of fitting a community of users to the best set of predictors. However, the data used to perform CF is sparse, and accuracy is limited by both the quantity and quality of information available. Mining the web has the potential to address these issues: the quality and quantity of ratings can be incremented by collecting external sources of rating information. In this work we introduce a method to perform CF with external data sources; furthermore, we show that a community of users can be partitioned according to what external source acts as a better predictor of each user's preferences. In particular, we find that a single kNN predictor can achieve remarkably high prediction accuracy if the data sources are selected optimally: designing a recommender system can thus be approached with the focus on data quality rather than algorithmic method.

1 Introduction

Recommender systems, based on collaborative filtering (CF), are displaying an evermore important and pervasive presence on the web. The problem of generating recommendations has been described as a *prediction* problem: based on a profile of user ratings, the system needs to predict future user ratings for other content in the future. The approaches adopted to perform CF can be broadly divided into two categories. The first are *statistical* approaches; these draw on the assumption of like-mindedness between users and therefore focus on a variety of classifiers that operate on the user-rating data; the most prominent candidates being based on matrix factorisation and neighbourhood methods [Koren, 2008][Herlocker *et al.*, 2004]. The second approach is based on *user modeling*; these methods augment statistical approaches by reasoning on the context and behaviours that emerge when people use recommender systems; recent examples include the rising interest in trust modeling for collaborative contexts, including [O'Donovan and Smyth, 2005].

Traditional CF suffers from the problem of *data sparsity*; the ability that a system has to make predictions for a user or item is limited by the lack of rating information. The data also has very high dimensionality; for example, the Netflix dataset[1] includes about half a million users and about twenty thousand movies. The mere size of the data implies that generating recommendations is a very expensive process that is difficult to scale to large communities. The current focus of much CF research is on improving the accuracy of the *algorithms* applied to generate recommendations. In particular, a number of successful statistical methods [Koren, 2008] combine an ensemble of predictors to produce higher accuracy. However, improving the classification method does nothing to improve the data that is being used when predicting user's preferences, and a fundamental limitating factor of any learning algorithm applied to the CF domain is the sparsity and potential inaccuracy of the data being used.

Similarly, algorithm-centric research also deters from fully modeling the implicit ways in which people form their opinions. While sociologists often model preference formation according to the principles of *homophily* (like-mindedness) and *social influence* (adopting the same preferences as influential members) [Axelrod, 1997][McPherson *et al.*, 2001], CF research has mainly centred its assumptions on the former theory. Although the task of identifying the source of influence in a set of user ratings seems daunting, this theory carries with it the assumption that there are a *range* of sources where users may form their opinions; in particular, not all users form their opinions by eliciting information from similar neighbours. The problem is thus how to model the way people form their opinions.

Mining the web for publicly available ratings has the potential to address the sparsity problem by drawing on the assumptions of *social influence*: there are a great number of online resources that contain a vast amount of ratings that may be accessed by users as they form their opinions. In this work we therefore propose to explore four different source datasets and evaluate the predictive power they have on a test set of user-movie ratings. Two of these source datasets were collected from the web, while the second two are de-

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

Dataset	Users	Ratings	Sparsity (%)
Flixster	77	585,293	79.02/0.01
Rotten Tomatoes	1,651	151,949	98.87
Netflix Training	9,980	1,432,259	99.19
Netflix Test	8,877	19,476	N/A

Table 1: Dataset Information

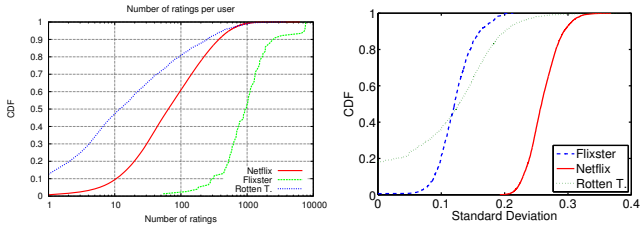


Figure 1: CDF of Ratings and Standard Deviation Per User

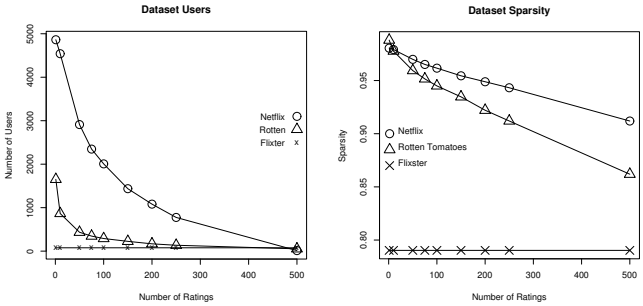


Figure 2: Number of users (left) and dataset sparsity (right) with rating threshold

rived from the a set of training data, based on *neighbours* and *power users*; Section 2 describes these datasets, and Section 3 highlights the statistical features that emerge between the sets. In Section 4 we introduce the method we implement to perform cross-dataset predictions, and Section 5 reports and analyses the results when each source dataset is used to make predictions on a common test set.

Our main result is that the accuracy of a CF prediction algorithms heavily depends on the *quality* of the information used to generate predictions, and the most appropriate source is user-dependent. In particular, matching users to the correct source of rating information has the potential to produce highly accurate recommendations when using a simple user-based kNN algorithm. We evaluate a number of benchmark methods that attempt to achieve this goal in Section 6; we thus introduce a novel perspective to CF, where the focus should not be so much on the *method* applied, but on the *data* that is used.

2 Information Sources

In this work we ran experiments using a subset of the Netflix prize data. Our subset consists of $10,000$ randomly selected Netflix users from the training set, and each of these user's probe ratings as a test set. To compliment this dataset we crawled two different sources of rating profiles: Rotten Tomatoes[2] and Flixster[3]. Based on how ratings are input into each of these systems, we call these sources *experts* and *enthusiasts* respectively:

Experts: The Rotten Tomatoes portal aggregates a number of cinema critic reviews from a wide range of web sources, including newspapers, specialized websites, and magazines. The critics use different rating scales; some range from 1-10 stars, others 1-5, and some use a 100-point scale. However, all of these ratings can be normalised. For example, a 9 out of 10 star rating is the same as 4.6 out of 5; we adopt a simple linear transpose to re-intepret ratings from one scale to another.

Enthusiasts: Flixster is one of the largest movie-oriented social networks, and therefore contains ratings given by the site's movie-enthusiast subscribers. We collected the profiles of the top-100 users from Flixster. However, not all users set their profiles to public: this reduced our collected dataset to 77 users. The Flixster users rate movies on a 1-5 star scale, but also have a further two options available: "want to see" (WS), and "not interested" (NI). In fact, the majority of ratings in the data fall into one of these two latter categories.

[2]http://www.rottentomatoes.com

[3]http://www.flixster.com/

Due to string-matching inconsistencies between the movie titles in Netflix and the crawled datasets, our datasets contain $6,088$ out of the $17,770$ available in Netflix; furthermore, to accomodate for this, we also cut any Netflix users who had no training ratings within this set of movies. A summary of the size of each dataset is given in Table 1. The analysis in Section 3 is based on this number of movies, which could be identified in *all three* datasets. We also compare the predictive performance of the external data sources to two other sets, each derived the the Netflix training subset we use:

Neighbours: The benchmark performance that we compare the above sources to is the approach adopted by traditional user-based kNN; there is no distinction made between users, who all come from the same community.

Power Users: This group is a subset of the "neighbours" group, the user profiles that, based a simple measure of profile size, are deemed to carry a significant amount of *reliable* information, and represent a sub sample of the community that may offer powerful predictions for the rest of the users. The idea of power users has been explored in the past [Cho *et al.*, 2007], and usually relies on identifying users based on a number of pre-defined heuristics. In this work we focus on profile size; that is, we assume that users who are proactively rating more items are following different behaviours to the casual rater [Herlocker *et al.*, 2004]. Note that all of the above groups strongly differ to results that would be obtained from clustering algorithms: users are grouped either based on profile attributes (rather than the value of their ratings), or based on where their ratings were crawled from. It is thus not guaranteed that users in the same group will agree with each other, whereas clustering algorithms tend to group users based on a notion of similarity.

3 Comparing Information Sources

In this section, we compare the three datasets. As introduced above, they can already be differentiated from one another according to a broad characterisation of the end-users of each system; however, in this section we examine the extent that ratings from different sources will differ in terms of summary statistics: the number and distribution of ratings, the sparsity, and rating deviation between per user.

Number of Ratings: With less than $1/20th$ of the users, the Flixster data contains over 5 times the number of ratings than the Rotten Tomatoes data. The same feature can be observed in Figure 1, which shows the cumulative distribution (CDF) of the number of ratings in each dataset. As the plot shows, 60% of the Rotten Tomatoes experts have about 30 or less ratings, 60% of the Netflix set users have 100 ratings or less, but the same proportion of Flixster reaches up to $1,000$ ratings.

Sparsity: Table 1 reports the sparsity values for each dataset; once again, Rotten Tomatoes and Netflix share similar sparsity values, while the Flixster dataset is a much denser set of ratings. The table also reports two separate sparsity values for the Flixster dataset. The first value, though excluding both WS and NI ratings, shows that the dataset is 79.02% sparse. Including all these extra ratings reduces the sparsity to 0.01%: in both cases, the user-rating matrix contains a much larger amount of ratings than Netflix alone. We also measured how the sparsity fluctuates as different sub samples of users are selected, reflecting the evaluation of *power users* we report in the Section 5. If we only select users who have rated more than α movies, both the number of users and resulting dataset sparsity will change. Figure 2 shows how these changes are affected by the rating threshold. The plots show that there is an uneven distribution of ratings amongst the users themselves, reinforcing the notion that users will behave differently as they interact with the system. The plots also confirm what was observed in Figure 1: the Netflix dataset, while having the highest number of users, also is also the sparsest of the three datasets.

Standard Deviation: Looking at this aspect aims to see the extent that each group of users agrees with each other by capturing the *spread* of ratings around each movie mean. As Figure 1 shows, the distribution of standard deviation values is very different from one community to the next. There are also a very small proportion of Flixster profiles that appear to be *outliers*: their profiles are full of the same rating for nearly *all* content. This causes the standard deviation over their ratings to be less than 0.05. These Flixster outliers will not be able to contribute useful information to any prediction, and can therefore be safely ignored.

4 Collaborative Filtering With External Data

There are a number of methods that can be implemented in order to use the ratings of the above datasets to predict the test set. In particular, one may simply combine all of the datasets into a single, larger training set that can be fed into any learning algorithm. However, in this work we aim to evaluate the potential that disparate sources have to predict a common test set: how well do *experts* predict the crowds? Do *enthusiasts* do better? In this light, and due to the semantics of cross-dataset prediction, we focus on a *single* method: the kNN algorithm.

The kNN can be built according to either the item-based or user-based paradigm. Both methods operate in very similar ways, and differ only in how they assume the underlying data is structured. Here we only consider the user-based approach. Once again, this makes our cross-dataset prediction highly explainable and transparent, which is a key aspect in building recommender systems that users trust [Herlocker *et al.*, 2004].

Implementing a kNN CF algorithm can be decomposed into three steps: (a) *neighbourhood formation*, where the top neighbours are computed for each user, (b) *rating aggregation*, where ratings for an item are collected and used to make a predicted rating, and finally (c) *recommendation and feedback*, where users update their profiles by responding to the recommendations they are given. When using external data sources, neighbours for a user in the training dataset are found from the source set; similarly, ratings for items in the test set will be predicted using the ratings in the selected source set. We identify neighbours using a *weighted* Cosine Similarity: the similarity $sim(a, b)$ between two users a and b is scaled according to how many items $N_{a \cup b}$ their profiles share in common. We base neighbour selection on a similarity threshold: any neighbour with similarity greater than zero is included. A predicted rating is then computed as a weighted average of deviations from each neighbour's mean [Herlocker *et al.*, 2004]. The only limitation we impose is a measure of prediction *confidence*: if less than 10 neighbour ratings have been found for a prediction, the prediction is set to the user mean. Setting the similarity threshold at zero and the confidence at 10 may not be optimal values for each dataset; in this work we focus on evaluating the ability to use adaptive *information sources* when generating predictions, rather than simply tweaking the *algorithm* itself for optimal performance.

We also noted that the Flixster dataset contains two additional ratings, NI and WS. It is not immediately transparent how these ratings should be transposed onto the scale in the target dataset, since they are difficult to place on an ordinal scale of ratings; however, a relationship between the NI rating and the movie average emerges in a few select cases. For example, consider two different movies that each have 1000 ratings. The first has a very high average, $4.8/5$, and only 10 NI ratings. The second has a very low average, $1.1/5$, and 800 NI ratings: it seems possible to assume that NI is roughly equivalent to a form of *negative feedback* provided by the user. However, we decided to ignore these ratings in the neighbourhood formation part of our algorithm.

We did test methods to include these ratings in the prediction step. Drawing from the assumption that NI may act as a form of *negative feedback* and WS represents a potential *positive opinion*, a Flixster neighbour n's NI and WS ratings for item i being predicted for user u (who has mean rating $\bar{r}_u$ and standard deviation σ_u) would map to:

$$NI_{n,u,i} = (\bar{r}_u - \sigma_u) \quad (1)$$

$$WS_{n,u,i} = (\bar{r}_u + \sigma_u) \quad (2)$$

Threshold (α)	# Users	RMSE	Proportion
0	9980	0.9700	22.82
50	5746	0.9709	9.09
100	3936	0.9722	9.26
200	2249	0.9748	8.81
300	1420	0.9778	9.33
400	928	0.9810	12.62
500	602	0.9854	28.08

Table 2: Power User Group RMSE Results

Source	RMSE	Proportions
Rotten Tomatoes	1.070	26.62
Flixster-WI/NS	1.207	25.25
Netflix	0.970	48.13
Flixster	1.259	N/A
User-Matched	0.856	N/A
Item-Matched	0.936	N/A
User-Item Matched	0.776	N/A

Table 3: RMSE When Predicting With External Sources & RMSE if users, items, and user-items were matched to the best source

Results when both using and ignoring these kind or ratings are reported in the following section.

5 Evaluation: Transparency of The RMSE

We measure the accuracy of predictions using the Root Mean Squared Error (RMSE) [Herlocker *et al.*, 2004]. We divide our experiments into two parts; in the first, we report the results when using different power user groups from within the Netflix dataset as sources. We then compare performance across the three source datasets.

Predicting With Power Users: Table 2 shows the relationship the rating threshold α used to define power users, the number of power users found who match this criteria, and the RMSE achieved when each group is used as the source set for predictions. The table shows that as the rating-threshold is increased, accuracy worsens. However, there are two points to note here: (a) as above, the algorithm has not been fully tuned for optimal performance (which is a dataset-dependent problem, subject to both the similarity metric and rating aggregation method implemented), and (b) relying on an aggregate error measure like the RMSE does not highlight the performance that is being achieved on a per-user basis. In other words, a single RMSE value does not show whether some users are better suited to certain information sources than others.

Based on the Table 2 alone, it seems that removing non-power users from the dataset results in a loss of prediction accuracy. To explore the veracity of this impression, we built a second matrix; in this case, each row corresponded to a user, and each column represented a source set of power users (according to a rating threshold). Each entry (i, j) in the matrix is the RMSE achieved on user i's test ratings with column j's source set. From this matrix, we were able to compute the proportion of users who best "affiliated" with each subset of power users. In other words, we could determine which source dataset was the most appropriate per user. Table 2 also shows the proportions of the $8,877$ test users dataset who affiliated best with each subgroup of power users. As the plot shows, there is in fact a very large proportion of users whose best source of predictions is the set of power users who have rated more than 500 movies. An algorithm which could manage to select the best power set group for each user would improve the aggregate accuracy from 0.97 to 0.914: users, therefore, associate differently with different sets of power users, and simply targetting a global optimal does not achieve the best possible accuracy. Conversely, it is also possible to infer from these experiments that the traditional nearest-neighbour model, based on selecting the *best* k neighbours for each user, is not optimal: improved accuracy is obtained when user groups are made apriori, and users then find their neighbours within those groups.

Predicting With External Data: The accuracy results when using all the external datasets are reported in Table 3. As the table shows, the overall accuracy when making predictions with an external sources is worse than simply using neighbours; from these values it would appear that exclusively using external data sources is not a viable option when designing a CF algorithm. The only point worth noting is that including the NI/WS ratings when making predictions using Flixster as a source provided an improvement. However, once again we constructed the user-data source RMSE matrix, and were able to extract the performance that a user-based kNN predictor would achieve if it were able to perfectly match users, items, or user-item pairs to the correct source data set. This time, each column of the error-matrix represented a different source set. The improvement is remarkable: assigning users to the correct data source provides (with this subsample of the data) an accuracy *below* the target of the Netflix prize. Repeating the above analysis across the three datasets also shows that there is no absolutely dominant source, the right column of Table 3 shows. The semantic interpretation of these results is that the Netflix dataset optimally predicts only half of the sampled population; movie critics and enthusiasts are more accurate for the others.

6 Benchmark Methods

Based on the above work it becomes apparent that classifiers like user-based kNN can achieve very high accuracy in the context of recommender systems if users are paired with the correct source of data; the main problem is thus how to infer, given a user profile, the correct source. Our first attempt considered various qualities of each user's profile, such as profile size, mean rating, rating standard deviation, and average agreement of the user's profile items with the movies' mean ratings. However, none of the individual components correlated strongly with each user's selection of optimal data source.

Our current work therefore focuses on how to infer what the best data source is for each user. In this work we propose and evaluate benchmark results derived from two methods. The first is based on *linear combinations* of the dataset

Type	Method	RMSE
Weight	Equal (1/3)	1.0215
Weight	Avg Similarity	0.9851
Weight	Group-Mean RMSE	1.0253
Weight	Training Set RMSE	1.013
Select	Max Avg. Similarity	0.9829
Select	Min Group-Mean RMSE	1.1243
Select	Best Training RMSE	1.208
Select	Most Training Confidence	0.9701

Table 4: Benchmark Method Performance

predictions, where a prediction of item i for user u is generated by each source, and the final prediction is computed as a *weighted* average of each score. The second method preclassifies each user to a particular dataset, and only computes one prediction per user-item pair using the *selected* dataset. This way, we can evaluate the method both in terms of the aggregate RMSE and the precision/recall metric related to how well the method paired each user with the appropriate dataset. The results we report here can be broadly categorised into two groups. The first are **structural properties**: We weight (or select) datasets based on emergent structural properties of the kNN algorithm; in particular, we measure the role that the similarity function plays when correlating a user to each set, by looking at the average positive similarity the users share with each source set. The second, **user-fit RMSE**, includes methods that weight (or select) the best source set according to how well each user *fits* the three sources, or how well each source predicts the user's training profile. This process entails a two-fold use of each user's training set of ratings; it is first used to compute similarity weights with members of each source set, and a second time to measure how well each source predicts the user's profile. The motivation for the latter group is as follows. Any given user u will have three potential neighbourhoods: Netflix (N), Rotten Tomatoes (RT), and Flixster (F). Each of these neighbourhoods will contain varying proportions of co-rated items with u's training set ratings, and the RMSE between these co-rated movies can be collected (ϵ_N, ϵ_{RT}, ϵ_F). Assuming that a relationship exists between each source's RMSE on the user's training profile and the predictive performance on the same user's test ratings, we can either *select* the source that provides the lowest RMSE, or weight the contribution of each source S proportionally to its accuracy on the training items:

$$w_S = \frac{(\Sigma_i \epsilon_i) - \epsilon_S}{\Sigma_i \epsilon_i} \tag{3}$$

Weighted Combinations We first tried a variety of linear combinations of each sources' predictions for each user's test items. As shown in Table 6, these ranged from weighting each source equally, to weighting each source according to the average shared similarity with the target user, weighting according to how well the target's profile fits the movie means generated from each source, and weighting according to how well each target fits the neighbourhood in each source. All the linear combinations of each sources' predictions failed to produce more accurate results on the test set than using the Netflix source alone. One of the primary reasons for this was that, in many cases, each source produced diverging predictions from the next: linear combinations of polarising predictions therefore hurt the overall results.

Group	Precision	Recall
Min Avg Similarity		
Netflix	0.457	0.867
Rotten Tomatoes	0.269	0.114
Flixster	0.277	0.018
Min Group-Mean RMSE		
Netflix	0.410	0.024
Rotten Tomatoes	0.261	0.546
Flixster	0.255	0.381
Min Training Set RMSE		
Netflix	0.307	0.002
Rotten Tomatoes	0.247	0.121
Flixster	0.263	0.834
Training Set Confidence		
Netflix	0.457	0.999
Rotten Tomatoes	0.0	0.0
Flixster	0.2	$8.2e^{-4}$

Table 5: User-Dataset Classification Performance

User-Source Classification The second set of experiments were performed in two steps. The first step assigns each user to a source by generating a mapping from each user to the the categorical set of sources, while the second step uses the mapping to generate predictions for each user's test set with the assigned source. As above, we tried classifying users according to how well they fit the source's mean ratings, their neighbourhood in each source, or by selecting the group that the user shares the highest amount of similarity with. We complimented these with a classifier that operated on how much *confidence*, or number of ratings, each source has about the target user's profile; the idea being that a user's behaviour mimics that of a particular source if both consistently rate the same items. Based on this methodology, we can measure two results: (a) the RMSE achieved on the test set after preclassifying each user, and (b) how well the pre-classification step works. We evaluated the latter based on the precision and recall metrics.

In this case, we find that the RMSE results are more encouraging: they differ from when only using the Netflix dataset by less than 0.012 using the average-similarity classification, and by 0.001 when the confidence-based classifier is implemented. However, exploring the precision and recall metrics in Table 6 highlights why these results were obtained: in the latter case, nearly *all* the users have been mapped to the Netflix source, thus producing the same results. Majority of the recall values are low, indicating a high proportion of misclassifications. Examining the results highlighted the fragility of the pre-classification step, and the dependence it had on the sources. In other words, users who were *wrongly* assigned to the Netflix source did not contribute as much error as those who were wrongly mapped to the smaller Flixster and Rotten Tomatoes datasets.

7 Related Work

The idea of using *experts* has been used before in CF research. The work by [Su *et al.*, 2007] defines experts as the *algorithms* that can be used to produce predictions; the authors construct a hybrid CF algorithm that outputs a weighted average of multiple CF algorithms. This significantly departs from the definition we apply here, where expertise is a quality of the *data* and not of the *method* applied to generate predictions using it. On the other hand, [Cho *et al.*, 2007] define experts as a subset of the users of a community based on a number of heuristics. In particular, expertise in an a domain is based on how many items a user has rated in that domain. This definition is closer to the way we identify *power users*, based on rating frequency, although we do not differentiate between domains within the items that can be rated.

Previous work [Aciar *et al.*, 2007] has also considered the problem of source selection; however, Aciar *et al* address the problems of identifying, selecting, and retrieving unstructured information from the web in order to produce recommendations. Sources are selected based on quantifiable relevance and considering how complete, diverse, and timely the data the sources contain is. The authors therefore propose a trust model to effectively select data sources. However, they adopt the broader goal of producing recommendations, while the work above centres on improving the accuracy of recommender system algorithms with a basic model of social influence. Examining how the quality of data relates to performance has also been discussed in the context of *computational trust*. In particular, [O'Donovan and Smyth, 2005] considers that users are more trustworthy sources of information if they tend to provide ratings that are good predictors of neighbour preferences. In our case we seek to identify the most trustworthy *source* of data per user in the Netflix community. Measuring trust based on a history of accurate predictions is similar to the baseline experiment in Section 6 that focused on how well users fits each source.

8 Conclusion

The primary motivation of this work was to highlight the dependence of CF algorithm's performance on the *quality* of the data that is being used to predict user preferences. We therefore explored the potential that a variety of datasets from the web have to predict a sample set of Netflix users. In doing so, we proposed a framework for cross dataset prediction, including methods to normalise data and interpret non-numeric ratings (NI and WS) on an ordinal scale. First, we examined the effect of learning to classify items based on a dense subset of the available training data, by extracting *power users* from the Netflix training set. We then analysed the predictive potential of *external* data sources, based on a collaborative method that generates a neighbourhood for a Netflix user composed of Flixster or Rotten Tomatoes profiles. We identified that the predictive power of both the power-user subsets and external sources is user-dependent; there are some users who are best predicted by power users, others by experts, enthusiasts, or neighbours. The two experiments, however, are not mutually exclusive. In fact, power users can also be identified and exploited within the Rotten Tomatoes dataset, and performing a further crawl of Flixster would supply the kNN algorithm with a richer set of enthusiast movie raters. The main focus of our future work will be combining the above results, in order to match to the best subset of a source dataset.

The potential of mining the web for rating information thus shifts the focus of building an accurate CF algorithm away from the *algorithm* and toward matching users to the appropriate *information* sources. The problem can thus be formulated as follows: given a user profile u, what profile *features* and emergent-*structural* properties of the kNN algorithm can be used to match the user to the best dataset? The preliminary experiments we report in Section 6 are promising, but still lack in the desired performance. In fact, alternative classification methods, with varying levels of dependence on the quality of the rating data, may perform better. We found that the strongest improvement was measured when data sets were adaptively selected for each user: the main result we observed is that classification accuracy is strongly related to the data source that is used, and improvement to the aggregate, global RMSE is proportional to how well users and data sources are linked. A viable option for building a CF system, therefore, need not rely on a combination of predictors [Koren, 2008], but rather on an optimal combination of data sources.

References

[Aciar *et al.*, 2007] S. Aciar, J. L. de la Rosa i Esteva, and J. L. Herrera. Information Sources Selection Methodology for Recommender Systems Based on Intrinsic Characteristics and Trust Measure . In *Proceedings of The 5th Workshop on Intelligent Techniques for Web Personalization*, Vancouver, Canada, 2007.

[Axelrod, 1997] R. Axelrod. The dissemination of culture: A model with local convergence and global polarization. *Journal of Conflict Resolution*, (41):203–226, 1997.

[Cho *et al.*, 2007] Jinhyung Cho, Kwiseok Kwon, and Yongtae Park. Collaborative filtering using dual information sources. *IEEE Intelligent Systems*, 22(3):30–38, 2007.

[Herlocker *et al.*, 2004] J. Herlocker, J. Konstan, L. Terveen, and J. Riedl. Evaluating collaborative filtering recommender systems. In *ACM TOIS*, volume 22, pages 5–53. ACM Press, 2004.

[Koren, 2008] Y. Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *ACM SIG KDD Conference*, 2008.

[McPherson *et al.*, 2001] M. McPherson, L. Smith-Lovin, and J.M. Cook. Homophily in social networks. *Annual Review of Sociology*, (27):415–444, 2001.

[O'Donovan and Smyth, 2005] J. O'Donovan and B. Smyth. Trust in recommender systems. In *IUI '05: Proceedings of ACM IUI*, pages 167–174. ACM Press, 2005.

[Su *et al.*, 2007] Xiaoyuan Su, Russell Greiner, Taghi M. Khoshgoftaar, and Xingquan Zhu. Hybrid collaborative filtering algorithms using a mixture of experts. In *Web Intelligence*, pages 645–649, 2007.