Recommender Systems

Papers and Notes from the 2001 Workshop
New Orleans, LA

In conjunction with ACM SIGIR Conference on
Research and Development in Information Retrieval

Jonathan L. Herlocker, Chair

Including the

2001 Who's Who in Recommender Systems

a snapshot of research and products related to recommender systems

Funding for these workshop notes provided by
ACM SIGIR Conference
Oregon State University

Organizing Committee

Jonathan L. Herlocker (Chair), *Oregon State University*
Joaquin Delgado, *Triplehop Technologies*
David McDonald, *FX Palo Alto Laboratory, Inc*
Douglas W. Oard, *University of Maryland*
Ian Soboroff, *National Institute of Standards and Technology (NIST)*

Notes from the Workshop Chair

The workshop notes are divided into two sections. The first section contains the research papers that were reviewed and accepted for presentation. The second section contains the *Who's Who in Recommender Systems*, where all scientists and practitioners in the area of recommender systems have been offered the opportunity to present a short description of their work in the recommender systems area.

All of the papers and notes found in these workshop notes can be found online at:

http://www.cs.orst.edu/~herlock/rsw2001

In addition, after the workshop has completed, a list of attendees will be posted.

CALL FOR PARTICIPATION
SIGIR 2001 Workshop on RECOMMENDER SYSTEMS
Thursday, September 13th
New Orleans, LA - USA
http://www.cs.orst.edu/~herlock/rsw2001/

Recommender systems assist and augment the transfer of recommendations between members of a community. A typical system collects preferences or opinions from individual users, then aggregates and transfers those recommendations to other members of the community. Recommender systems are often referred to as "collaborative filtering systems".

In recent years, interest in recommender systems has dramatically increased, driven primarily by demand for Internet personalization applications. Many major e-commerce web sites are currently using recommender systems to personalize their content layout and target sales. Most notably, the largest online retailer Amazon.com makes extensive use of recommender system technology.

There is also considerable potential for recommender systems to make a difference off the Internet as well. As one example, recommender systems are an important part of knowledge management solutions on corporate intranets - notably in recommending expertise. Another example is the Tivo digital video recorder appliance, which has demonstrated that recommender systems can benefit those who have never used a computer or the Internet.

This recommender workshop is a follow up to the extremely successful recommender systems workshop that was hosted by SIGIR 1999 in Berkeley, CA.

WORKSHOP GOALS

In this workshop, we seek to bring together researchers and practitioners who are involved in the development of recommender systems. The theme will be "algorithms, applications, and interfaces". The key goals of the workshop are to

1) Provide a forum where current practitioners can meet and discuss recommender system issues
2) Identify the most important open problems
3) See presentations of important work or results related to recommender systems

PLANNED ACTIVITIES & SELECTION PROCESS

The workshop will include moderated discussions, panels, and presentations of recent research. While some paper presentations will be included, the workshop will be primarily focused around providing opportunities for discussion.

ATTENDANCE & PARTICIPATION REQUIREMENTS

There are three ways to participate in the 2001 Workshop on Recommender Systems.

1. We invite any individual or group involved in the research, development, or production of recommender systems to submit a one page summary of their recommender systems related work. These will be aggregated and published. See the "WHO'S WHO IN RECOMMENDER SYSTEMS" section below for more info. No attendance is required!
2. If you plan to attend the workshop, please submit a summary as described in 1. However, attendees may extend their summaries to two pages. Please also include an indication of your intent to attend.

3. In addition, if you have recent research results related to recommender systems that you would like to present at the workshop, you can submit a research paper. Papers will be reviewed as described below. Accepted papers will be published in the workshop notes.

Attendance at the workshop is limited to 50 people. Attendees will be accepted on a first-come, first-served basis. Don't wait until the last minute, as the '99 SIGIR recommender workshop filled up entirely.

The workshop will include a number of presentations of current research work and results. Those who would like time to present their research should include with their position statement a research paper that is suitable for publication (up to 12 pages). Those research papers that are accepted will be published in the workshop notes.

Position papers of potential presenters will be reviewed by the organizing committee. Papers will be selected with the following considerations:

1) Quality of work
2) Impact of work
3) Applicability of work to the interests of workshop attendees
4) Diversity of presentations (given that a limited number of presentation slots will be available, we want to ensure that no two presentations are overly similar)

WHO'S WHO IN RECOMMENDER SYSTEMS

As part of this recommender system, we are inviting all individuals or groups who are involved with recommender systems to submit summaries of their past or current efforts. All of these summaries will be aggregated and distributed, both through the workshop notes and on the web. The intent of this effort is to take a snapshot of the recommender systems community in research, development, and production. The goal is to increase the awareness of all those involved with recommender systems. By participating in this "who's who in recommender systems", you ensure that your work in this area will be recognized.

Non-attendees are invited to submit a one-page summary, attendees two pages. We would prefer to receive only one summary per institution or company, although multiple summaries will be accepted from the same institution or company if the authors work independently and the summaries describe significantly different work.

Summaries should include the names of members associated with the work, contact information, and a web URL, if appropriate.

Summaries will not be peer reviewed, although we reserve the right to refuse summaries that we feel are inappropriate. Summaries will not be proof-read or edited - they will be published as submitted.

Full-Length Presentations

Compositional Recommender Systems Using Case-Based Reasoning Approach

Stefano Aguzzoli* and **Paolo Avesani** and **Paolo Massa**
ITC-irst - 38050 Povo, Italy
{aguzzoli,avesani,massa}@itc.it

Abstract

We introduce an application combining CBR and collaborative filtering techniques in the music domain. We describe a scenario in which a new kind of recommendation is required suitable to summarize in one suggestion many recommendations. Our claim is that to recommend a set of goods is different from recommending many times a single good. The paper illustrates how a case-based reasoning approach can provide an effective solution to this problem. Moreover the proposed framework that extends collaborative filtering for compositional recommendation systems allows us to reduce the drawbacks related to the user profiles.

1 Introduction

The notion of intelligent sales support [4; 15] is going to be widely recognized as a new challenge for e-commerce applications. The new scenario promoted by the Internet has increased the opportunities to customize products; now, it is possible to deliver goods or services close as much as possible to a user's needs.

The pre-sales phase in the context of electronic commerce is being revisited, because the role of the customer has changed. In the past, a customer could only select a product from a set catalogue made in advance by the supplier organization. Now, the customer is directly involved in the process of assembling the product, a task that requires the combination of a set of predefined components in accordance with user preferences.

The music market [9] represent one of the most notable scenarios in which this evolution is occurring. In the past, the delivered goods were compact discs containing compilations (that is, selected collections) of audio tracks; nowadays, it is possible to buy individual tracks rather than entire precompiled CDs. In spite of this change, the notion of "compilation" is still relevant, because e-commerce systems enable the purchase of single tracks, but the delivery still relies on compact discs as media. In this scenario, the role of editor of compilations has moved from the market expert to the final user.

New Web portals devoted to the sale of sound tracks[1] allow each user to create his own compilation. Usually, the assembling process is based on an iterative step that adds a sound track selected from the result set of a query to a repository of music.

From this perspective, the advantage introduced by the personalization of the product could bring some drawbacks. Firstly, the simple iterative step of adding one track after another quickly becomes boring, as an average compilation includes some 15 tracks. Secondly, query engines on track repositories are not able to provide querying over many important features of music, because tracks tagging is time consuming and user dependent. Finally, there is no way to formulate a query to a repository in which each individual track is enriched with information on what is an appropriate context for its inclusion in a compilation.

Giving the users the opportunity to assemble their own compilation, enables the portal to collect important knowledge about the sound tracks. Indeed, every single compilation may be considered as an instance of the *genre of use* related to its sound tracks. This kind of genres are not the usual categories used to classify music (rock, jazz, classical, and so on); the genres of use represents all the different perspectives along with a user can look at music. Following this approach, one can imagine to have the standard compilations of "rock" rather than "jazz music", but at the same time one could have compilations of "driving west coast" music rather than "a night with my girl" music, which may possibly put together tracks that would be kept separated under a traditional classification system.

From this perspective, a repository of compilations may be considered a powerful case base to support both the editing of personalized compilations and the detection of the genres of use introduced by customers.

In the following, we examine how case-based reasoning can be applied to support product personalization [17;

*Current address: Computer Science Department, University of Milan; email: stefano@gongolo.usr.dsi.unimi.it.

[1] See as examples the following Web sites: www.mixfactory.com, www.musicalacarte.net.

16; 18] by taking advantage of collaborative filtering techniques [7; 13]. After presenting the motivations for this work, we briefly introduce CoCoA, a recommendation system for music compilations. The following section gives a detailed explanation of the design of the case-based retrieval engine used by CoCoA, which is based on the collaborative filtering approach. We then show the results of an empirical evaluation carried out on a dataset derived from EachMovie [11] and on Medline, one of the benchmark of the information retrieval community. The paper concludes with a brief discussion of the results, including a comparison with the Singular Value Decomposition technique [8; 6].

2 Motivations

As described above, the customer is involved in the process of assembling their own compilation, by adding one track after another. Some of the Web portals that distribute music have started to enhance this interaction by providing a recommendation utility. By selecting the recommendation option, the customer can access a list of tracks that the system has detected as being relevant for the current partial compilation. Even though this kind of systems relies on repositories of compilations, there is no way for the customer to browse them.

In the recommendation approach, the suggestion (the sound track) is provided without the context (the compilation) that could help the user to receive a sort of explanation of why the track proposed by the system has been selected. Furthermore, the customer cannot directly interpret and reuse what other customers have done in the past by looking at the existing compilations and deriving a new one from them.

Our claim is that to recommend a set of tracks differs from recommending one track many times. Let us consider a situation where a partial compilation under development by the current user includes an ambiguity over the target genre of use: for example "jazz" or "funky". In this case recommendations concerning tracks from both of these genres of use are valid. But the suggestions may not be effective if the goal is to enable the choice of the missing tracks at once. The user in such a case is not aware of the real genres of the tracks and can't perform a consistent multiple choice without making a mistake. If instead the user interacts iteratively with the recommender facility after the choice of one track, for example a track related to the "jazz" genre, then he or she can obtain a more focussed recommendation.

The first motivation of our work was to introduce the CBR loop in the context of compositional e-commerce, where a customer is required to collect a set of components to fulfill the pre-sales phase. Our thesis is that the collected use cases offer a viable solution for partitioning recommended tracks into meaningful collections to support the completion task. The following sections gives a simple introduction to CBR and the two related interaction models designed for CoCoA.

A further motivation underlying our work is to integrate into the CBR loop a retrieval engine so that there is no need to address the content description in order to compute the similarity measure. In the domain of music this aspect is crucial for two reasons: firstly, because the definition of music categories does not meet a general agreement; and secondly, because the features that usually describe a sound track do not discriminate them between the different genres of use.

The integration of CBR and Collaborative Filtering in the context of compositional e-commerce presents a couple of drawbacks. The first is related to the choice of the notion of a case: we cannot associate a case description to a single user, because the experience is represented by a single compilation. Each compilation can be considered as an episode where the goal of collecting a set of tracks following a given criteria has been achieved. Looking at one compilation as a single case is one of our working assumptions. At the same time, from a collaborative filtering perspective, we consider the single compilation as the profile of a virtual user. The second drawback is related to the notion of rating. Usually the collaborative filtering techniques rely on the correlation between two user profiles where the ratings allow the system to assess their similarity. In the case of compilations, an explicit rating of the sound tracks is not available and we can simply assume that all the tracks belonging to the same compilation have an equal rating.

Summarizing, we have to cope with the following two drawbacks:

1. The case is a single compilation, its dimension is fixed and very small compared to the set of all tracks. Compare this with the profile of the movies graded by a user. The information in this latter profile grows indefinitely as the user continues to grade newly seen movies. In the compilation scenario, the tracks are given once and for all, and two compilations are not likely to have any significant overlap of common tracks.

2. The case does not contain any graded rating of the tracks occurring within it: put simply, a track is either there or not there. We do not know from the case if a track is actually significant for the case or mildly important or even just to be regarded as noise. Moreover, all we know of a track is its name.

Given the assumptions illustrated above, we have to deal with the problem of making the collaborative filtering approach effective in a situation where drawbacks 1 and 2 are present. The objective of this work is to extend the standard collaborative filtering approach to enable its use in a CBR loop in the context of compositional e-commerce.

Before introducing our approach to achieving this objective, we briefly illustrate the CoCoA system, which represents the first practical application of our work.

Figure 1: **CoCoA GUI:** the CoCoA graphical user interface.

3 Case-Based Reasoning

Case-based reasoning (CBR) is a problem solving paradigm based on the 4-R loop: retrieve, reuse, revise, retain [1]. Starting from the assumption that similar problems share similar solutions, the general cycle of CBR, given a new problem formulation, begins with the retrieval of the most similar past cases, continues with the reuse of the knowledge encoded in the proposed cases to revise the past solutions, and finishes wuth the retention of the new problem solving experience as potentially useful information for future situations.

Each step involves different techniques to deal with representational choices, learning rates, efficiency issues and effectiveness measures.

Although the reuse step makes the main difference between CBR and information retrieval the original contribution of this work involves the retrieval step, with the combining of a case based approach and collaborative filtering techniques. Let us now consider how a recommender system can be designed by taking into account the CBR perspective when looking at the application.

4 The CoCoA System

CoCoA, a COmpilation COmpiler Advisor, has been designed as a Web service that supports the editing of a compilation by means of a repository of sound tracks and a case base of compilations.

The system architecture allows the sound track providers to plug this compositional facility into their existing sales software. The management and distribution of the sound tracks is not part of CoCoA, which supports only the assembling process and the management of the case base repository, i.e. the database of virtual compilations, containing only track indexes.

A running version of this Web service has been employed for the Karadar[2] Web site, a provider of free classical music. Starting from the catalogue of Karadar (a set of around 2000 sound tracks) CoCoA allows the user to collect a set of tracks in a compilation, while the download of the final product is left again to the Karadar Web site.

The bootstrap of the case base has been performed by looking at the log files of the Web site. The mining of this kind of data allowed for the detection of the sequences of downloads concerning 5 to 12 sound tracks. For each of these sequences, a compilation was stored in the case base.

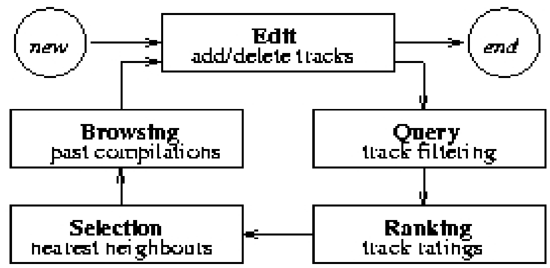


Figure 2: **User Adaptation:** the CBR interaction supports the retrieval of past similar compilations, the choice concerning the reuse of a such kind of information remains in charge of the user.

From a CBR perspective, the system implements two models of interaction: the recommend model and the complete model. The recommend model is depicted in figure 2, which summarizes the main steps of the interaction. The user looking at the repository of sound tracks can iteratively delete or add new tracks to his own compilation. On demand, the user can decide to invoke the recommend utility in order to add new tracks taken from past compilations. First, the current partial compilation is addressed to the retrieval engine as an implicit query where the terms are the sound tracks currently included in the collection. A second step performs a ranking process to sort the compilations of the case base with respect to the given query. A third step selects the closest compilations, which are presented to the user. A fourth step, under the control of the user, allows the browsing of the compilations returned by the query; the selection of tracks for the completion of the user's collection.

A second interaction model, namely the completion model, is essentially organized as the previous one. The main difference concerns a supplementary step between the selection and the browsing steps. A prediction process enable the system to perform a synthesis of new compilations providing a possible completion of the partial collection assembled by the user. In this case, the user will browse a set of "virtual" compilations, since the case base does not necessarily include any of them. Of

[2] See the URL `http://www.karadar.com`.

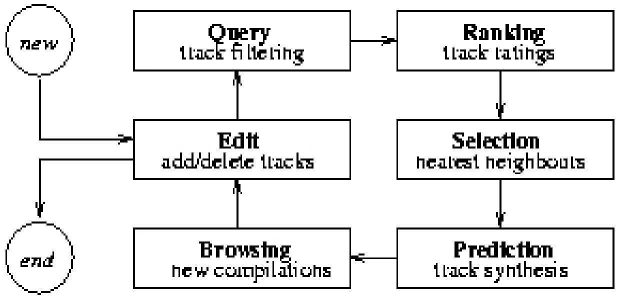


Figure 3: **System Completion:** the CBR steps of retrieval and reuse are managed automatically by the system; the revision and the refinement of proposed solution are still in charge of the end user.

course, the completion of the final collection still remains the decision of the user.

After having presented the interaction models, we now explain how a recommendation engine supporting this kind of model has been designed, and how the drawbacks highlighted in the previous section have been overtaken.

5 The recommendation engine

The recommendation engine represents the core of a CBR system for compositional e-commerce. It takes as inputs the partial collection being assembled and the repository of past compilations. Its output is a subset of the case base and a ranking of the selected compilations.

To illustrate in detail how, given these inputs, we obtain a sound output, let us go through a step-by-step definition of the basic elements of the computational method. For the sake of simplicity, we continue to refer to the music domain, without loss of generality.

Definition 1 (Sound Tracks) *$T = \{t_i\}$ is a finite collection of components, namely sound tracks.*

It is worthwhile to notice that the definition of sound tracks does not require to provide a feature value specification. Any individual sound track is represented only by its id code.

Definition 2 (Track Compilation) *Let n be some fixed integer much smaller than the cardinality of T. Then $C = \{c_i = \{t_{i1}, \ldots, t_{in}\} \mid t_{ij} \in T\}$ is a finite collection of sound tracks.*

Although a compilation entails the notion of sequence, in the following we will not take this information into account. From now on, we assume that the compositional problem is concerned only with the notion of collection.

Definition 3 (Active Compilation) *$A = \{a_i = \{t_{i1}, \ldots, t_{im}\} \mid t_{ij} \in T,\ m \leq n\}$ is a finite collection of sound tracks at most of the same cardinality of a track compilation.*

The active compilation plays the role of the recommendation query when a collection of track compilations is organized in a case base. Before introducing how a single compilation can be viewed as a user profile from a collaborative filtering perspective, let us see what kind of information can be extracted from a case base.

Throughout this paper by CB we shall denote the subset of C forming the current case base.

Definition 4 (Frequency of track pairs) *Let $s : 2^T \rightarrow 2^C$ be a function that given a subset S of sound tracks returns the subset of compilations that include them: $s(S) = \{c \in CB | S \subseteq c\}$. Then $f_\nu : 2^T \rightarrow [0,1] \subset \mathcal{R}^+$ allows to obtain the frequency of a subset of sound tracks: $f_\nu(S) = |s(S)||CB|^{-1}$. An approximation of the sound track probability can be derived from the track frequency:*

$$\hat{P}(t_i) = f_{t_i}(t_i) = f_\nu(\{t_i\}) \quad f_{t_i}(t_j) = \frac{|s(\{t_i, t_j\})|}{|CB|} = f_{t_j}(t_i)$$

Using this formulation we can derive also the approximation of the conditional probability:

$$\hat{P}(t_i|t_j) = \frac{f_{t_j}(t_i)}{f_{t_j}(t_j)} = \frac{f_{t_i}(t_j)}{f_{t_j}(t_j)}$$

The above definition introduced the functions $f_{t_i}(t)$ that can be considered a kind of track profile where $f_{t_i}(t_j)$ says how much the track t_i "likes" the track t_j, i.e. the rating of t_j formulated by t_i.

Now, we need to summarize a kind of compilation profile starting from the $f_{t_i}(t)$ that belongs to a given compilation.

Definition 5 (Compilation Synthetic Function) *Given a function $\mu : 2^T \rightarrow [0,1]^T$ and a compilation $c \in CB \subset C$, where $c = \{t_1, \ldots, t_n\}$, a characteristic function can be defined as:*

$$f_c : T \rightarrow [0,1], \quad f_c(t) = \mu(t_1, \ldots, t_n)(t)$$

The function $f_c(t)$ allows to summarize a compilation profile that does not restrict its definition to the subset of track components. $f_c(t)$ is defined over all T; it does mean that we can interpret the value $f_c(\bar{t})$, where $\bar{t} \notin c$, as an estimate of how much the compilation c likes the track $\bar{t}$. In this way we can cope with the scarce overlap of compilations (see drawback 1). Further, the range of any f_c is the interval $[0,1]$ thus providing the needed graduation of the ratings (see drawback 2).

In the following, we propose an hypothesis for μ although we are aware that its choice is related to a definition of a loss function. More details on the evaluation criteria will be illustrated in the section dedicated to the experiments.

Definition 6 (μ Function) *Let μ be a linear composition:*

$$\mu(d_1, \ldots, d_n)(t) = \sum_{d_j} \kappa(d_j) \frac{f_{d_j}(t)}{f_{d_j}(d_j)}$$

where $\kappa(d_j)$ is a weight to balance the contribution of the single components of the given compilation.

Although the objective of the parameter $\kappa(d_j)$ is to measure how much a track is representative for the genre of use of the compilation which it belongs to, for the sake of simplicity in the following we assume $\kappa(d_j)$ to be a constant.

Once a synthetic function to model the compilation is available, we can simply rely on a linear process to assess the similarity and to rank the case base.

Definition 7 (Selection Process) *A function $f_r : A \times 2^C \to 2^C$ to select the nearest neighbours compilations, given a distance measure d and a threshold k:*

$$f_r(a_i, CB) = \{c_i | d(a_i, c_i) \leq k,\ c_i \in CB,\ CB \subset C\}$$

To compare two compilations as usually performed in collaborative filtering, we can refer to the Pearson correlation coefficient. This measure can now be effective, because the drawbacks 1 and 2 described earlier have been removed. The compilation profiles are widely defined over T increasing their overlap. Moreover the compilation profiles, even the implicit and tacit rating derived from the compilation, can differentiate the agreement on different sound tracks through the synthesis of frequency pairs.

Definition 8 (Pearson Correlation Coefficient) *Let $a \in A$ be an active compilation and $c \in CB$ a past compilation, given a range $D \subseteq T$ a function $d : A \times CB \to [-1, 1]$ returns the linear correlation factor between two compilations:*

$$d(a, c) = \frac{\sum_{t \in D}(\mu_a(t) - \bar{\mu}_a)(\mu_c(t) - \bar{\mu}_c)}{\sqrt{\sum_{t \in D}(\mu_a(t) - \bar{\mu}_a)^2 \sum_{t \in D}(\mu_c(t) - \bar{\mu}_c)^2}}$$

where $\bar{\mu}_a$ and $\bar{\mu}_c$ respectively denote the mean value of μ_a and μ_c over D.

Associated with the Pearson correlation coefficient an algorithm has been developed, called *agave-t* , implementing the schema illustrated above. The major enhancement introduced by this algorithm is that two compilations can be compared even if they do not share any track. Moreover, the comparison can occur also between sound tracks that do not belong to any of the two compilations, which are the majority.

We developed a different version of the algorithm, *agave-c* , restricting the range of comparison to the subset of tracks belonging to the active compilation or the track compilation. This restriction reduces the problem of dimensionality suffered by the approach based on user profiles. For the same reason, we have designed a third version of the algorithm, *agave-cvs* , that implements a different comparison based on vector similarity.

Definition 9 (Vector Similarity Coefficient) *Let $a \in A$ be an active compilation and $c \in CB$ a past compilation, given a range $D = c \cup (a \cap T)$ a function $d : A \times CB \to \mathcal{R}^+$ return the cosine of the angle between two compilations:*

$$d(a, c) = \sum_{t \in D} \frac{\mu_a(t)}{\sqrt{\sum_{t \in D} \mu_a(t)^2}} \frac{\mu_c(t)}{\sqrt{\sum_{t \in D} \mu_c(t)^2}}$$

The variation introduced with the *agave-c* algorithm has a significant impact on the computational complexity. The restriction on the dimensionality of D allows us to reduce the complexity to a linear dependency on the size of the case base, because in $O(|D||CB|)$ the term $|D|$ becomes a constant factor.

Definition 10 (Completion Process) *Let $a \in A$ be an active compilation and $c \in CB$ a past compilation, a function $f_s : A \times CB \to C$ returns a compilations $\hat{c}$ such that $a \cap \hat{c} = a$ and $\hat{c} \cap \{a \cup c\} = \hat{c}$.*

A trivial interpretation of the completion process could be a function that, while preserving the tracks defined in the active query, performs the synthesis of a new compilation by a random selection of the missing tracks.

6 Related Works

Recently, many works have been published in the area of Collaborative Filtering, but the focus of our paper is slightly different. While in standard recommendation systems the goal is to predict with accuracy the affinity of the user to a specific good, in the context of compositional e-commerce this goal is extended to a collection of goods. For this reason a direct comparison of the proposed approach with the main recommendation system is not possible.

We prefer to move the analysis of the related work to a basic technique underlying many applications in the area of information retrieval and collaborative filtering: the singular value decomposition (SVD) [2; 14; 3; 12]. Although other methods have been derived from SVD, such as the latent semantic analysis (LSI) [5] and the related probabilistic variation (PLSI) [10], we limit the comparison to SVD because it is representative of the approaches based on dimensionality reduction.

SVD can be viewed as a technique for deriving a set of uncorrelated indexing variables or factors (and these can be our *genres of use*); each track and compilation is represented by its vector of factor values. Note that by virtue of the dimension reduction, it is possible for compilations with different tracks to be mapped into the same vector of factor values.

A byproduct of the SVD approach is the possibility to estimate a correlation factor between two collections that do not share any of their components. This feature is strongly related to the challenge engaged with the compositional e-commerce.

Let us summarize how the SVD works looking at our chosen domain of music. The starting point of SVD is a very sparse matrix X of $c \times t$ (compilations $\times$ tracks) where every cell contains 1 if that track is contained in that compilation and 0 otherwise. SVD allows to decompose this rectangular matrix into three other matrices of a very special form (the resulting matrices contain "singular vectors" and "singular values").

These special matrices yield a breakdown of the original relationships into linearly independent components or factors. In general, many of these components are very small, and may be ignored, leading to an approximate model with many fewer dimensions. In this reduced model all the track-track, compilation-compilation and track-compilation similarity is now approximated by values in this smaller number of dimensions. The result can be still represented geometrically by a spatial configuration in which the dot product or cosine between vectors representing two objects corresponds to their estimated similarity.

A comparison with SVD has been accomplished on Medline, an information retrieval dataset that contains 1033 abstracts from the National Library of Medicine and 30 queries, both already represented as list of terms and their frequency.

Although the task usually performed on such a dataset is a little different we tried to look at the documents as a collection of keywords, without taking into account their occurence frequency. Moreover we didn't consider setting a threshold to filter the results set, this is the reason why we don't measure the recall.

Table 1 illustrates the precision results computed on medline for *agave-t* and SVD for different k dimensions. As a baseline the answer that maximizes the overlap of the keywords between documents and queries has also been computed (named *rif*).

It is worth noting that although the optimization of the dimensionality reduction for SVD, *agave-t* achieve good results in precision. The next section shall go much more in detail concerning with the experimentation of the proposed approach.

	agave-t	*rif*	*svd200*	*svd30*
precision	0.563	0.547	0.467	0.313

Table 1: The results on Medline dataset where the size of the result set has been fixed equal to ten.

7 Evaluation results

After the presentation of our approach, based on the mix of CBR and CF, and a description of the state of the art concerning the basic technique for implementing a recommendation system, we now focus our attention to the empirical analysis.

In the following, we illustrate how we selected the dataset to perform a comparative analysis. We then explain how we have designed the experiments and the evaluation setup. Finally, in a discussion of the results we try to summarize the main contribution of this work.

7.1 Dataset setup

To evaluate our approach we needed information about compilations of objects (music tracks, movies, etc...). The CoCoA system is going to be deployed on the Web; in the meantime, we cannot exploit the case base that will be available as soon as it achieves a significant number of users. Waiting for a real case base of compilations, we had to look for an alternative solution.

The community that is working on the recommendation system provides some dataset based on the notions of user profiles and goods. But the concept of *compilation* is slightly different from *profile* because the former is an aggregation of items usually with a small and limited size, while the latter is allowed to grow indefinitely. A user profile is effective as long as it increases, and very often the recommendation accuracy is related to its size.

Moreover, a further difference distinguishes a user profile from a compilation: all the items of a collection are strongly related by a sort of homogeneous factor. We can assume that, even if not explicit, it does exist a common feature that takes the same value for all the items of the collection. We are actually assuming that every compilation has a rationale for its existence, even if this rationale is not explicitly stated or even known. On the contrary, a user profile does not provide such a kind of information concerning the relationship among a selection of items rated by a single user.

For these reasons, we could not use a traditional Collaborative Filtering dataset, so we have designed a specific one for our purpose. The synthesis of the dataset has been derived starting from Eachmovie [11], a dataset composed by 2811983 ratings expressed by 72916 users on 1648 movies. A related dataset, the IMDB, supports an extended content description for every movie like the year of release and the indication of the genres.

We generated a dataset of movie collections giving a genre to each of them. A genre has been defined following a criteria of uniformity along one or more features for every movie of the collection. We defined 15 different genres: before 90', 90'-93', 94', 95', 96', Action, Animation, Art_Foreign, Classic, Comedy, Drama, Family, Horror, Romance, Thriller. This choice allows that the same movie belongs to at least a couple of collection genres.

To simulate the way in which people select the movies to aggregate them in collections, we performed a pseudo-random extraction using the data of Eachmovie. We derived the distribution from the frequency of the rated movies. Also, the number of compilations of each genre is proportional to the number of ratings people gave to movies belonging to that genre.

The total number of compilations is 2995 made out of 1086 movies and each compilation contains 10 movies.

7.2 Setup experiment

The task selected for the experiment is the retrieval of the 10 nearest neighbours among the train set compilations for every test set compilation. We have created 10 crossfolds dividing randomly the 2995 compilations in a train set of 2395 compilations and a test set of 600.

To evaluate the interaction model illustrated above, we needed to test the usual way of invoking the recommendation utility: the typical scenario sees the user insert some movies (or items, in the general case) and then ask for similar past compilations, hopefully of the genre sketched in the active compilation. For this reason it is crucial to properly manage short queries intended as partial compilations.

To test this aspect, we have designed different experiments on the same dataset, in which we changed the shadowed portion of every test compilation. More precisely, we replicated the standard test step many times taking into account respectively only 3, 6 and 9 movies of each compilation in the test set. In this way we can simulate the evolution of the active compilation and the different complexity of the related query. The challenge is to achieve good accuracy performance overall when only a sketch of the compilation is available, i.e. the system is dealing with "short" queries.

The performance evaluation measures the accuracy in terms of percentage of recommended compilations with the same genre of the active one. For every test compilations we have extracted the 1nn. The computation of the mean error over the 10nn has given similar results.

7.3 Discussion of results

In figure 4 you can see our algorithm *agave-t* compared to SVD with different number of kept dimensions. Although the parameter for the dimensionality reduction has been optimized, namely $k = 200$, *agave-t* still outperforms SVD. This result provides an evidence that the proposed approach is an effective technique to reduce the sparseness coefficient that passes from 99% to 25%.

Anyway it is worth noticing that the improvement introduced with *agave-t* is not balanced by the on line computational effort. The theoretical online complexity, crucial for time performance, gives opposite results: $O(k|CB|)$ for SVD and $O(|T||CB|)$ for *agave-t* . This comparison is quite crucial because in general k is much smaller than $|T|$.

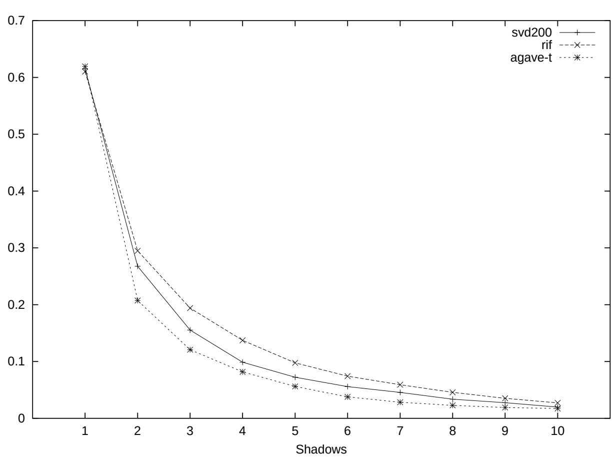


Figure 4: Mean error of *genre of use* matching between test compilation and the 1nn train compilation recommended. The experiments are relatives to different shadow sizes.

However, assuming it is viable to slightly relax the accuracy constraint to give much more attention to the time response, the proposed approach can achieve interesting results. Taking this perspective we also have tried a version of the algorithm, called *agave-c* , that reduces the computation of the Pearson correlation coefficient between the test and train compilations considering only the movies of those compilations instead of all the movies. In this way, the mean complexity is reduced to $O(2\overline{c}|CB|)$ where $\overline{c}$ is the average number of movies in a compilation (10, in our experiments). As expected, the accuracy error increases but, differently from the previous case, the enhancement achieved on the time response is much more significant than the loss of accuracy. Moreover the accuracy of SVD decreases much faster than *agave-c* . To highlight this aspect we have compared the achieved results with *svd15* that has a comparable online complexity. The results concerning *agave-c* are depicted in figure 5.

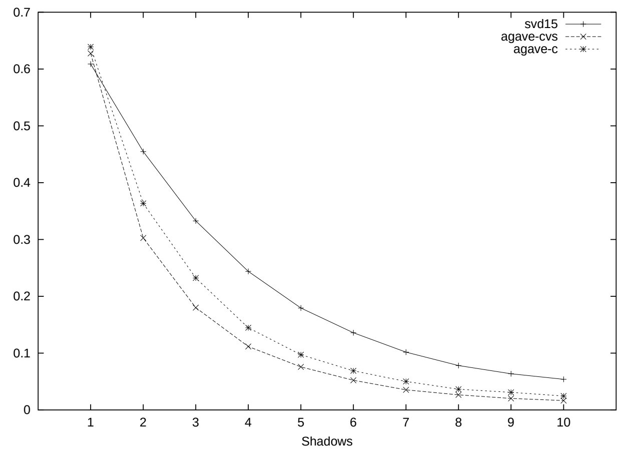


Figure 5: Mean error of *genre of use* matching between test compilation and the 1nn train compilation recommended. The experiments are relatives to different shadow size. The algorithms compared are *agave-c* *agave-cvs* and *svd15*. The loss in accuracy of *agave-c* compared to *agave-t* is counter-balanced by a meaningful improvement of the time response.

The plot shows also *agave-cvs* , a version of *agave-c* in which, to compute the similarity between two compilations, we substitute Pearson correlation coefficient with the cosine of the angle of the vectors representing the two compilations.

This simpler measure (used for instance by SVD) works even better. The reason is related to the dimensionality reduction: the reduced extent prevents the problem of the curse of dimensionality. In this case the Pearson correlation coefficient is less effective because the notion of sparseness does not apply: let us remind the reader that on average we have to compare two vectors of 10-15 components.

As previously stated, our main goal is to reach a certain level of accuracy especially when trying to find a compilation to recommend in reply to a query containing few movies. The results show that, even in presence of partial query compilations, the accuracy of the recommended compilation is good. This is significant especially when compared to Collaborative Filtering algorithms that work well only after the acquisition of a meaningful history interaction to achieve a rich user profile.

From the point of view of CBR this result allows to exploit the collaborative filtering technique without the restriction of the user profile but enabling the association between case and a single user interaction. In this way the case base becomes the repository of the interaction episodes and the user is free to play different roles, i.e. to assemble compilations not necessarily related to each other.

8 Future directions

First of all, we hope as soon as possible to make available a real case base of compilations through the deployment of CoCoA[3].

This step is the premise for the refinement of the evaluation setup that has to overcome the constraint of fixed genres. We need to deal with a situation where every single compilation might be representative of a new genre of use. The final goal is to have the opportunity to assess the genre proximity.

The hypothesis for $\mu_c(t)$ illustrated in this paper represents a preliminary solution to the aggregation problem of the different track profiles encoded by $f_i(t_j)$. A deeper understanding concerning alternative schemas has to be achieved. For example: how to balance the contribution of the different track profiles is still an open problem. We are already taking into account graph-based clustering techniques to detect how much a single track can be considered representative of a genre of use.

Finally we have not yet designed the completion function to support the related interaction model. The delivery of support of this kind could create the premises for an interesting comparison from a CBR point of view between the recommender and the completion models.

9 Acknowledgment

We would like to thank Anna Perini, Michele Nori, Gianluca Mameli and Marco Grimaldi for their important role in the design and the development of CoCoA.

The *Each-Movie* database was generously provided by the System Research Center of Digital Equipment Corporation, which makes it available for use in research.

References

[1] A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues, methodological variations, adn system approaches. *AI Communications*, 7(1):39–59, 1994.

[2] M. Berry and S. Dumais. Using linear algebra for intelligent information retrieval. *SIAM Review 37(4)*, pages 573–595, 1995.

[3] D. Billsus and M. J. Pazzani. Learning collaborative information filters. Technical report, AAAI, July 1998.

[4] Padraig Cunningham, Ralph Bergmann, S. Schmitt, R. Traphoener, and S. Breen. Websell: Intelligent sales assistants for the world wide web. Technical report, Trinity College Dublin, 2000.

[5] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, pages 391–407, 1990.

[6] George E. Forsythe, Michael A. Malcolm, and Cleve B. Moler. *Computer Methods for Mathematical Computations.* Prentice-Hall, Englewood Cliffs, NJ 07632, USA, 1977.

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

[8] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press and North Oxford Academic, Baltimore, MD, USA and Oxford, England, 1983.

[9] C. Hayes and P. Cunningham. Smart radio: Building music radio on the fly. In *Expert Systems 2000, Cambridge, UK*, 2000.

[10] Thomas Hofmann. Probabilistic latent semantic analysis. pages 177–196, 2001.

[11] P. McJones. Eachmovie collaborative filtering data set, dec systems research center, 1997. http://research.compaq.com/SRC/eachmovie/.

[12] M.H. Pryor. The effects of singular value decomposition on collaborative filtering. Senior Honors Thesis PCS-TR98-338, Dartmouth College, June 1998.

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

[14] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Application of dimensionality reduction in recommender system – a case study. In *ACM WebKDD 2000 Web Mining for E-Commerce Workshop*, 2000.

[15] J.B. Schafer, J. Konstan, J., and Riedl. Recommender systems in e-commerce. In *Proceeding of the ACM Conference on Electronic Commerce*, Pittsburgh, PA, USA, November 1999.

[16] S. Schmitt and R. Bergmann. Applying case-based reasoning technology for product selection and customization in electronic commerce environments. In *12th Bled Electronic Commerce Conference*, 1999.

[17] Armin Stahl and Ralph Bergmann. Applying recursive CBR for the customization of structured products in an electronic shop. In *EWCBR*, pages 297–308, 2000.

[3] A beta version is available at the following address `http://cocoa.itc.it/cocoakaradar/`

[18] Wolfgang Wilke, Mario Lenz, and Stefan Wess. Intelligent sales support with CBR. In *Case-Based Reasoning Technology*, pages 91–114, 1998.

Content-Boosted Collaborative Filtering

Prem Melville, Raymond J. Mooney and Ramadass Nagarajan
Department of Computer Sciences
University of Texas
Austin, TX 78712
{melville,mooney,ramdas}@cs.utexas.edu

ABSTRACT

Most recommender systems use Collaborative Filtering or Content-based methods to predict new items of interest for a user. While both methods have their own advantages, individually they fail to provide good recommendations in many situations. Incorporating components from both methods, a hybrid recommender system can overcome these shortcomings. In this paper, we present an elegant and effective framework for combining content and collaboration. Our approach uses a content-based predictor to enhance existing user data, and then provides personalized suggestions through collaborative filtering. We present experimental results that show how this approach, *Content-Boosted Collaborative Filtering*, performs better than a pure content-based predictor, pure collaborative filter, and a naive hybrid approach. We also discuss methods to improve the performance of our hybrid system.

1. INTRODUCTION

Recommender systems help overcome information overload by providing personalized suggestions based on a history of a user's likes and dislikes. Many on-line stores provide recommending services e.g. Amazon, CDNOW, BarnesAndNoble, IMDb, etc. There are two prevalent approaches to building recommender systems — Collaborative Filtering (CF) and Content-based (CB) recommending. CF systems work by collecting user feedback in the form of ratings for items in a given domain and exploit similarities and differences among profiles of several users in determining how to recommend an item. On the other hand, content-based methods provide recommendations by comparing representations of content contained in an item to representations of content that interests the user.

Content-based methods can uniquely characterize each user, but CF still has some key advantages over them [11]. Firstly, CF can perform in domains where there is not much content associated with items, or where the content is difficult for a computer to analyze — ideas, opinions etc. Secondly a CF system has the ability to provide serendipitous recommendations, i.e. it can recommend items that are relevant to the user, but do not contain content from the user's profile. Because of these reasons, CF systems have been used fairly successfully to build recommender systems in divers domains [9, 19]. However they suffer from two fundamental problems:

- *Sparsity*
  Stated simply, most users do not rate most items and hence the user-item rating matrix is typically very sparse. Therefore the probability of finding a set of users with *significantly* similar ratings is usually low. This is often the case when systems have a very high item-to-user ratio. This problem is also very significant when the system is in the initial stage of use.
- *First-rater Problem*
  An item cannot be recommended unless a user has rated it before. This problem applies to new items and also obscure items and is particularly detrimental to users with eclectic tastes.

We overcome these drawbacks of CF systems, by exploiting content information of the items already rated. Our basic approach uses content-based predictions to convert a sparse user ratings matrix into a full ratings matrix; and then uses CF to provide recommendations. In this paper, we present the framework for this new hybrid approach, Content-Boosted Collaborative Filtering (CBCF). We apply this framework in the domain of movie recommendation and show that our approach performs significantly better than both pure CF and pure content-based systems.

The remainder of the paper is organized as follows. Section 2 provides an illustrative example to motivate our approach. In Section 3, we describe our domain and the gathering of data. Section 4 describes in detail our implementation of the content-based predictor, the CF algorithm and the hybrid approach. We present our experimental results in Section 5 and explain why our system performs well in Section 6. Section 7 proposes methods to improve our CBCF predictions. In Section 8, we discuss prior attempts at integrating collaboration and content; and finally in Section 9, we conclude with some future extensions to our work.

2. MOTIVATING EXAMPLE

In this section, we describe a common scenario in recommender systems and show why both pure collaborative and content-based methods fail to provide good recommendations. We take the domain of movie recommendations as a representative case.

In most systems, users provide feedback on items that they liked or disliked, using which profiles are formed to learn about the specific interests of each user. For example, in

User A	User B
2001: Space Odyssey	Exorcist
ET	Evil Dead
	Star Wars
	Contact

Table 1: Typical user profiles: movies liked

movie recommendations, typical user profiles could be as shown in Table 1. The table shows two users A and B and their profiles that consists of movies that each liked. Pure CF systems try to find *neighbors*(similar users) for a user by computing similarity measures based on the common set of movies that two users rated. If there is no overlap in the movies of two users, they will not be considered as neighbors. Thus in this example, A and B are not neighbors and potentially movies that B liked, may not be recommended to A even though their profiles suggest that both like science fiction movies.

Pure content-based systems on the other hand, form profiles for each user independently. Thus a typical system would learn that A likes science fiction movies, while B likes both horror and science fiction movies. But since each user is considered separately, movies that do not share any content with the ones already rated will not be considered for recommendation. In this case, horror movies that B liked may not be recommended to A, even though A and B seem to have a common taste — science fiction and it is quite likely that A will like horror movies as well.

Clearly both of the above approaches are inadequate. Let us consider a different approach. We could use a content-based system to predict A's preferences. A content-based predictor would rate *Star Wars* and *Contact* highly based on A's predilection for science fiction. Now if we were to perform CF using A's content-based predictions, A and B would appear similar; and subsequently B's preferences would be recommended to A. Our CBCF predictor is based on this approach.

3. DOMAIN DESCRIPTION

We demonstrate the working of our hybrid approach in the domain of movie recommendation. We use the user-movie ratings provided by the EachMovie dataset and the movie details from the Internet Movie Database (IMDb) [1, 2]. We represent the content information of every movie as a set of slots (features). Each slot is represented simply as a bag of words. The slots we use for the EachMovie dataset are: movie title, director, cast, genre, plot summary, plot keywords, user comments, external reviews, newsgroup reviews, and awards.

3.1 EachMovie Dataset

The EachMovie dataset is provided by the Compaq Systems Research Center, which ran the EachMovie recommendation service for 18 months to experiment with a collaborative filtering algorithm. The information they gathered during that period consists of 72,916 users, 1,628 movies, and 2,811,983 numeric ratings. To have a quicker turn-around time for our experiments, we only used a subset of the EachMovie dataset. This dataset contains 7,893 randomly selected users and 1,461 movies for which content was available from IMDb. The reduced dataset has 299,997 ratings for 1,408 movies. The average votes per user is approximately 38 and the sparsity of the user ratings matrix is 2.6%.

The dataset provides optional unaudited demographic data such as age, gender, and the zip code supplied by each person. For each movie, information such as the name, genre, release date and IMDb URL are provided. Finally, the dataset provides the actual rating data provided by each user for various movies. User ratings range from zero-to-five stars. Zero stars indicate extreme dislike for a movie and five stars indicate high praise.

3.2 Data Collection

The content information for each movie was collected from the Internet Movie Database (IMDb). A simple crawler follows the IMDB link provided for every movie in the EachMovie dataset and collects information from the various links off the main URL. We presently download content such as plot summary, plot keywords, cast, user comments, external reviews (newspaper or magazine articles), newsgroup reviews, and awards. This information, after suitable preprocessing such as elimination of stop words etc., is collected into a vector of bag of words, one bag for each feature describing the movie.

4. SYSTEM DESCRIPTION

The general overview of our system is shown in Figure 1. The web crawler uses the URLs provided in the EachMovie dataset to download movie content from IMDb. After appropriate preprocessing, the downloaded content is stored in the Movie Content Database. The EachMovie dataset also provides the user-ratings matrix; which is a matrix of users versus items, where each cell is the rating given by a user to an item. We will refer to each row of this matrix as a *user-ratings vector*. The user-ratings matrix is very sparse, since most items have not been rated by most users. The content-based predictor is trained on each user-ratings vector and a pseudo user-ratings vector is created. A pseudo user-ratings vector contains the user's actual ratings and content-based predictions for the unrated items. All pseudo user-ratings vectors put together form the pseudo ratings matrix, which is a full matrix. Now given an active user's[1] ratings, predictions are made for a new item using CF on the full pseudo ratings matrix.

Sections 4.1 and 4.2 describe our implementation of the content-based predictor and the pure CF component. In Section 4.3 we describe our hybrid approach in detail.

4.1 Pure Content-based Predictor

To provide content-based predictions we treat the prediction task as a text-categorization problem. We view movie content information as text documents, and user ratings 0-5 as one of six class labels. We implemented a bag-of-words naive Bayesian text classifier [15] to learn a user profile from

[1]The active user is the user for which predictions are being made.

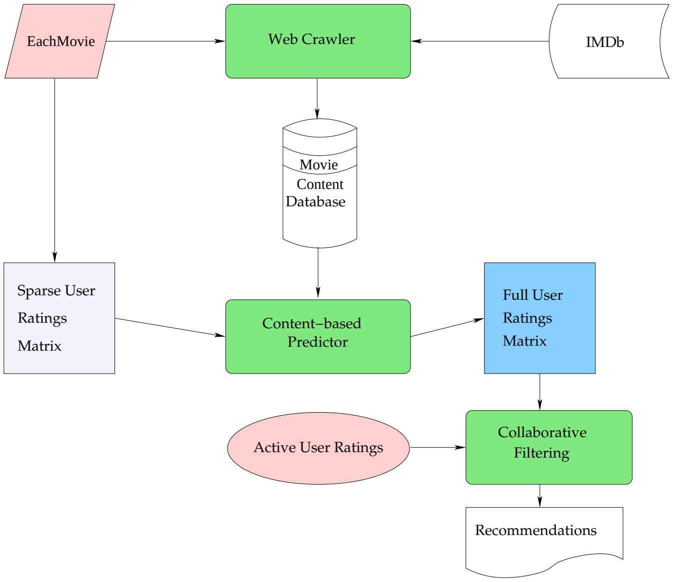


Figure 1: System Overview

a set of rated movies i.e. labeled documents. A similar approach to recommending has been used effectively in the book-recommending system *LIBRA* [16, 17].

We use a multinomial text model [14], in which a document is modeled as an ordered sequence of word events drawn from the same vocabulary, V. The naive Bayes assumption states that the probability of each word event is dependent on the document class but independent of the word's context and position. For each class c_j, and word (token), $w_k \in V$, the probabilities, $P(c_j)$ and $P(w_k|c_j)$ must be estimated from the training data. Then the posterior probability of each class given a document D, is computed using Bayes rule:

$$P(c_j|D) = \frac{P(c_j)}{P(D)} \prod_{i=1}^{|D|} P(a_i|c_j)$$

where a_i is the ith word in the document, and $|D|$ is the number of words in the document. The prior $P(D)$ can be ignored, since it is a constant for any given document.

In our case, since movies are represented as a vector of "documents", d_m, one for each slot (where s_m denotes the mth slot), the probability of each word given the category and the slot, $P(w_k|c_j, s_m)$, must be estimated and the posterior category probabilities for a film, F, computed using:

$$P(c_j|F) = \frac{P(c_j)}{P(F)} \prod_{m=1}^{S} \prod_{i=1}^{|d_m|} P(a_{mi}|c_j, s_m)$$

where S is the number of slots and a_{mi} is the ith word in the mth slot. The class with the highest posterior probability determines the predicted rating.

The model parameters are estimated using the algorithm in Algorithm 1. Note that Laplace smoothing [12] is used to avoid zero probability estimates. The evaluation of the content-based recommender can be found in the appendix.

Algorithm 1 Training the Content-Based Predictor

$Train\_Naive\_Bayes(Examples, C)$

Each example in *Examples* is a vector of bag-of-words and a category corresponding to a 0-5 rating. Each bag of bag-of-words corresponds to a slot e.g. title, cast, reviews, etc. C is the set of all possible categories. This function estimates the probability terms $P(a_{mi}|c_j, s_m)$, describing the probability that a randomly drawn word from a slot s_m in an example in class c_j will be the word a_{mi}.

1. Calculate class priors, $P(c_j)$
   - $docs_j \leftarrow$ subset of documents from *Examples* for which the class label is j
   - $P(c_j) \leftarrow \frac{|docs_j| + \frac{1}{|Examples|}}{|Examples| + \frac{|C|}{|Examples|}}$
2. Calculate conditional probabilities, $P(a_{mi}|c_j, s_m)$

   For each slot s_m,
   - $Vocabulary_m \leftarrow$ set of all distinct tokens occurring in slot s_m in all examples
   - For each possible class c_j
     - $Text_{mj} \leftarrow$ a single document created by concatenating all bags-of-words appearing in slot s_m and in class c_j
     - $n \leftarrow$ total number of distinct word positions in $Text_{mj}$
     - For each token, a_{mi} in $Vocabulary_m$
       * $n_k \leftarrow$ number of times token a_{mi} occurs in $Text_{mj}$
       * $P(a_{mi}|c_j, s_m) \leftarrow \frac{n_k + \frac{1}{|Examples|}}{n + \frac{|Vocabulary_m|}{|Examples|}}$

4.2 Pure Collaborative Filtering

We implemented a pure collaborative filtering component that uses a *neighborhood-based algorithm* [11]. In neighborhood-based algorithms, a subset of users are chosen based on their similarity to the active user, and a weighted combination of their ratings is used to produce predictions for the active user. The algorithm we use can be summarized in the following steps:

1. Weight all users with respect to similarity with the active user.
   - Similarity between users is measured as the Pearson correlation between their ratings vectors.
2. Select n users that have the highest similarity with the active user.
   - These users form the *neighborhood*.

3. Compute a prediction from a weighted combination of the selected neighbors' ratings.

In step 1, similarity between two users is computed using the Pearson correlation coefficient, defined below:

$$P_{a,u} = \frac{\sum_{i=1}^{m}(r_{a,i} - \overline{r}_a) \times (r_{u,i} - \overline{r}_u)}{\sqrt{\sum_{i=1}^{m}(r_{a,i} - \overline{r}_a)^2 \times \sum_{i=1}^{m}(r_{u,i} - \overline{r}_u)^2}} \qquad (1)$$

where $r_{a,i}$ is the rating given to item i by user a; and $\overline{r}_a$ is the mean rating given by user a.

In step 3, predictions are computed as the weighted average of deviations from the neighbor's mean:

$$p_{a,i} = \overline{r}_a + \frac{\sum_{u=1}^{n}(r_{u,i} - \overline{r}_u) \times P_{a,u}}{\sum_{u=1}^{n} P_{a,u}} \qquad (2)$$

where $p_{a,i}$ is the prediction for the active user a for item i. $P_{a,u}$ is the similarity between users a and u. n is the number of users in the neighborhood. For our experiments we used a neighborhood size of 30, based on the recommendation of [11].

It is common for the active user to have highly correlated neighbors that are based on very few co-rated (overlapping) items. These neighbors based on a small number of overlapping items tend to be bad predictors. To devalue the correlations based on few co-rated items, we multiply the correlation by a *Significance Weighting* factor [11]. If two users have less than 50 co-rated items we multiply their correlation by a factor $sg_{a,u} = n/50$, where n is the number of co-rated items. If the number of overlapping items is greater than 50, then we leave the correlation unchanged i.e. $sg_{a,u} = 1$.

4.3 Content-Boosted Collaborative Filtering

In content-boosted collaborative filtering, we first create a *pseudo user-ratings vector* for every user u in the database. The pseudo user-ratings vector, v_u, consists of the item ratings provided by the user u, where available, and those predicted by the content-based predictor otherwise.

$$v_{u,i} = \begin{cases} r_{u,i} : & \text{if user } u \text{ rated item } i \\ c_{u,i} : & \text{otherwise} \end{cases}$$

In the above equation $r_{u,i}$ denotes the actual rating provided by user u for item i, while $c_{u,i}$ is the rating predicted by the pure content-based system.

The pseudo user-ratings vectors of all users put together gives the dense pseudo ratings matrix V. We now perform collaborative filtering using this dense matrix. The similarity between the active user a and another user u is computed using the Pearson correlation coefficient described in Equation 1. Instead of the original user votes, we substitute the votes provided by the pseudo user-ratings vectors v_a and v_u.

4.3.1 Harmonic Mean Weighting

The accuracy of a pseudo user-ratings vector computed for a user depends on the number of movies he/she has rated. If the user rated many items, the content-based predictions are good and hence his pseudo user-ratings vector is fairly accurate. On the other hand, if the user rated only a few items, the pseudo user-ratings vector will not be as accurate. We found that inaccuracies in pseudo user-ratings vector often yielded misleadingly high correlations between the active user and other users. Hence to incorporate confidence (or the lack thereof) in our correlations, we weight them using the *Harmonic Mean weighting* factor(*HM weighting*, for short).

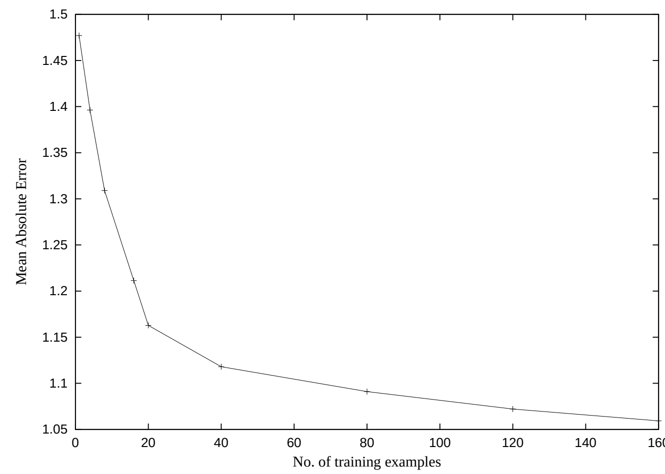


Figure 2: Learning Curve for the Content-based Predictor

$$\begin{aligned} hm_{i,j} &= \frac{2m_i m_j}{m_i + m_j} \\ m_i &= \begin{cases} \frac{n_i}{50} : \text{if } n_i < 50 \\ 1 : \text{otherwise} \end{cases} \end{aligned}$$

In the above equation, n_i refers to the number of items that user i has rated. The harmonic mean tends to bias the weight towards the lower of the two values — m_i and m_j. Thus correlations between pseudo user-ratings with at least 50 user-rated items each, will receive the highest weight, regardless of the actual number of movies each user rated. On the other hand, even if one of the pseudo user-rating vectors is based on less than 50 user-rated items, the correlation will be devalued appropriately.

The choice of the threshold 50 is based on the learning curve[2] of the content predictor. As can be seen in Figure 2, initially as the predictor is given more and more training examples the prediction performance improves, but at around 50 it begins to level off. Beyond this is the point of diminishing returns; as no matter how large the training set is, prediction accuracy improves only marginally.

To the HM weight, we add the significance weighting described in Section 4.2, and thus obtain the *hybrid correlation weight* $hw_{a,u}$.

$$hw_{a,u} = hm_{a,u} + sg_{a,u} \qquad (3)$$

[2] The appendix provides a detailed explanation of the generation of the learning curve.

4.3.2 Self Weighting

Recall that in CF, a prediction for the active user is computed as a weighted sum of the mean-centered votes of the best-n neighbors of that user. In our approach, we also add the pseudo active user[3] to the neighborhood. However, we may want to give the pseudo active user more importance than the other neighbors. In other words, we would like to increase the confidence we place in the pure-content predictions for the active user. We do this by incorporating a *Self Weighting* factor in the final prediction:

$$sw_a = \begin{cases} \frac{n_a}{50} \times max & : \text{if } n_a < 50 \\ max & : \text{otherwise} \end{cases} \quad (4)$$

where n_a is the number of items rated by the active user. Again, the choice of the threshold 50 is motivated by the learning curve mentioned earlier. The parameter max is an indication of the over-all confidence we have in the content-based predictor. In our experiments, we used a value of 2 for max.

4.3.3 Producing Predictions

Combining the above two weighting schemes, the final CBCF prediction for the active user a and item i is produced as follows:

$$p_{a,i} = \overline{v}_a + \frac{sw_a(c_{a,i} - \overline{v}_a) + \sum\limits_{\substack{u=1 \\ u \neq a}}^{n} hw_{a,u} P_{a,u}(v_{u,i} - \overline{v}_u)}{sw_a + \sum\limits_{\substack{u=1 \\ u \neq a}}^{n} hw_{a,u}}$$

In the above equation $c_{a,i}$ corresponds to the pure-content predictions for the active user and item i. $v_{u,i}$ is the pseudo user-rating for a user u and item i and $\overline{v}_u$ is the mean over all items for that user. sw_a, $hw_{a,u}$ and $P_{a,u}$ are as shown in Equations 4, 3 and 1 respectively; n is the size of neighborhood. The denominator is a normalization factor that ensures all weights sum to one.

5. EXPERIMENTAL EVALUATION

In this section we describe the experimental methodology and metrics we use to compare different prediction algorithms; and present the results of our experiments.

5.1 Methodology

We compare CBCF to a pure content-based predictor, a CF predictor, and a naive hybrid approach. The naive hybrid approach takes the average of the ratings generated by the pure content-based predictor and the pure CF predictor. For the purposes of comparison, we used a subset of the ratings data from the *EachMovie* data set (described in Section 3.2). Ten percent of the users were randomly selected to be the test users — all test user had rated at least forty movies. From each user in the test set, ratings for 25% of items were withheld. Predictions were computed for the withheld items using each of the different predictors.

The quality of the various prediction algorithms were measured by comparing the predicted values for the withheld ratings to the actual ratings.

[3]Pseudo active user refers to the pseudo user-ratings vector based on the active user's ratings.

5.2 Metrics

The metrics for evaluating the accuracy of a prediction algorithm can be divided into two main categories: *statistical accuracy metrics* and *decision-support metrics*. Statistical accuracy metrics evaluate the accuracy of a predictor by comparing predicted values with user-provided values. To measure statistical accuracy we use the mean absolute error (MAE) metric — defined as the average absolute difference between predicted ratings and actual ratings. In our experiments we computed the MAE on the test set for each user, and then averaged over the set of test users.

Decision-support accuracy measures how well predictions help users select *high-quality* items. We use Receiver Operating Characteristic (ROC) sensitivity to measure decision-support accuracy. A predictor can be treated as a filter, where predicting a high rating for an item is equivalent to accepting the item, and predicting a low rating is equivalent to rejecting the item. The ROC sensitivity is given by the area under the ROC curve — a curve that plots *sensitivity* versus 1-*specificity* for a predictor. Sensitivity is defined as the probability that a good item is accepted by the filter; and specificity is defined as the probability that a bad item is rejected by the filter. We consider an item *good* if the user gave it a rating of 4 or above, otherwise we consider the item *bad*. We refer to this ROC sensitivity with threshold 4 as ROC-4. ROC sensitivity ranges from 0 to 1, where 1 is ideal and 0.5 is random.

Herlocker et al. used the same metrics to compare their algorithms [11]. The statistical significance of any differences in performance between two predictors was evaluated using two-tailed paired t-tests [15].

5.3 Results

Algorithm	MAE	ROC-4
Pure content-based predictor	1.059	0.6376
Pure CF	1.002	0.6423
Naive Hybrid	1.011	0.6121
Content-boosted CF	**0.962**	**0.6717**

Table 2: Summary of Results

The results of our experiments are summarized in Table 2 and Figure 3. As can be seen, our CBCF approach performs better than the other algorithms on both metrics. On the MAE metric, CBCF performs 9.2% better than pure CB, 4% better than pure CF and 4.9% better than the naive hybrid. All the differences in MAE are statistically significant ($p < 0.001$).

On the ROC-4, metric CBCF performs 5.4% better than pure CB, 4.6% better than pure CF and 9.7% better than the naive hybrid. This implies that our system, compared to others, does a better of job of recommending high-quality items, while reducing the probability of recommending bad items to the user.

Interestingly, *Self Weighting* did not make significant improvements to our predictions.

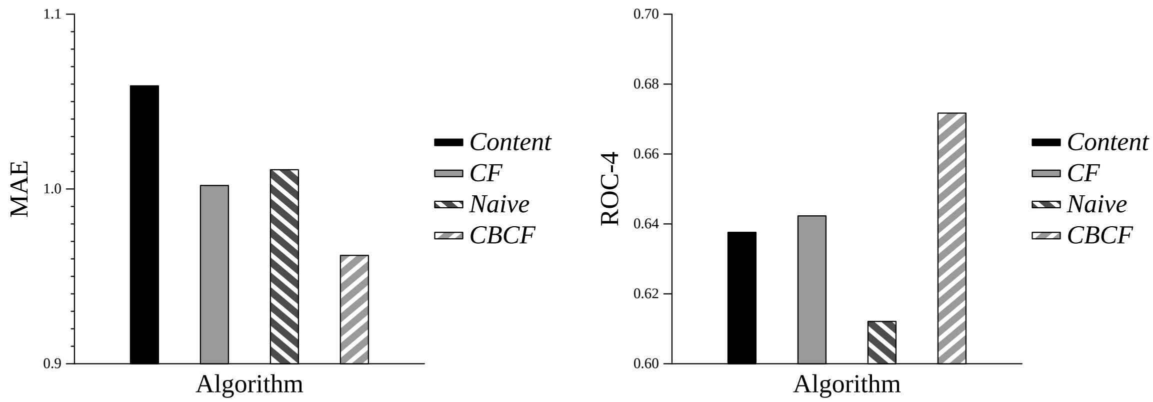


Figure 3: Comparison of algorithms

6. DISCUSSION

In this section we explain how content-boosted collaborative filtering overcomes some of the shortcomings of pure CF; and we also discuss some of our performance results.

6.1 Overcoming Sparsity and the First-Rater Problem

Since we use a pseudo ratings matrix, which is a full matrix, we eliminate the root of the sparsity and first-rater problems. Pseudo user-ratings vectors contain ratings for all items; and hence all users will be considered as potential neighbors. This increases the chances of finding similar users.

The original user-ratings matrix may contain items that have not been rated by any user — there are 53 such movies in our dataset. In a pure CF approach these items would be ignored. However in CBCF, these items would receive a content-based prediction from all users. Hence these items can now be recommended to the active user, thus overcoming the first-rater problem.

6.2 Finding Better Neighbors

A crucial step in CF is the selection of a neighborhood. The neighbors of the active user entirely determine his predictions. It is therefore critical to select neighbors who are most similar to the active user. In pure CF, the neighborhood comprises of the users that have the best n correlations with the active user. The similarity between users is only determined by the ratings given to co-rated items; so items that have not been rated by both users are ignored. However, in CBCF, the similarity is based on the ratings contained in the pseudo user-ratings vectors; so users do not need to have a high overlap of co-rated items to be considered similar. Our claim is that this feature of CBCF, makes it possible to select a better, more representative neighborhood. For example, consider two users with identical tastes who have not rated any items in common. Pure collaborative filtering would not consider them similar. However, pseudo user-ratings vectors created using content-based predictions for the two users would be highly correlated, and therefore they would be considered neighbors. We believe that this superior selection of neighbors is one of the reasons that CBCF outperforms pure CF.

6.3 Making Better Predictions

As discussed above, CBCF improves the selection of neighboring users. In traditional CF, we would compute a prediction for each item as a weighted sum of only the *actual* ratings of these neighbors. However, in our approach, if the actual rating from a neighboring user does not exist, we use his content-based predicted rating. This approach is motivated by the hypothesis that if a user is highly correlated to the active user then his content-based predictions are also very relevant to the active user. We believe that the use of the content-based ratings of neighbors to compute predictions is another important factor contributing to CBCF's superior performance.

6.4 Self Weighting

Content predictions based on a large number of training examples tend to be fairly accurate, as is apparent from Figure 2. Hence, giving a greater preference to such predictions should improve the overall accuracy of our hybrid prediction. Interestingly, this was not reflected in our results. This may because of the choice of the max parameter in Equation 4, which was fixed to be 2 in our experiments. A higher value for max would increase the weight of content-based predictions, and might yield better results.

6.5 Naive Hybrid

The naive hybrid approach that we used to compare our system with was inspired by [6]. We found that this approach was a poor strawman to compare with. As can be seen by the results the naive hybrid performs worse than CF on the MAE metric. It also performs poorly on the ROC-4 metric, when compared to the other approaches. In Section 8, we present some other approaches we can use as benchmarks to compare our approach against.

7. IMPROVING CBCF

Due to the nature of our hybrid approach, we believe that improving the performance of the individual components would almost certainly improve the performance of the whole system. In other words, if we improved our pure content-based predictor or the CF algorithm, we would be able to improve our system's predictions. A better content-based predictor would mean that the pseudo ratings matrix generated would more accurately approximate the actual *full* user-ratings matrix. This in turn, would improve the chances of finding more representative neighbors. And since the final predictions in our system are based on a CF algorithm, a better CF algorithm can only improve our system's performance. We discuss some methods we could use to improve the individual components.

7.1 Improving the Content-based Predictor

In our current implementation of the content-based predictor, we use a naive Bayesian text-classifier to learn a six-way classification task. This approach is probably not ideal, since it disregards the fact that classes represent ratings on a linear scale. For example, the posterior probabilities for the classes 1 and 3 might be 0.4 and 0.6 respectively, this would imply that a good prediction should be close to 2. But the classifier will predict a 3 i.e the class with the higher posterior probability.

This problem can be overcome by using a learning algorithm that can directly produce numerical predictions. For example, logistic regression and locally weighted regression [7] could be used to directly predict ratings from item content. We should be able to improve our content-based predictions using one of these approaches.

7.2 Improving the CF Component

The CF component in our system can be improved by using a *Clustered Pearson Predictor* (CPP) [8], instead of the Simple Pearson Predictor (SPP) that we currently employ. The CPP algorithm creates k clusters of users based on the k-means clustering algorithm. A profile is created for each cluster, which contains the average of the ratings given for each item by all the users in the cluster. Now, predictions are computed using SPP where only the k profiles generated earlier are considered as potential neighbors. Fisher et al. claim that this approach is more accurate than SPP [8]. CPP also has the advantage of being more scalable than SPP.

8. RELATED WORK

There have been a few other attempts to combine content information with collaborative filtering. One simple approach is to allow both content-based and collaborative filtering methods to produce separate ranked lists of recommendations, and then merge their results to produce a final list [6]. There can be several schemes to merging the ranked lists, such as interleaving content and collaborative recommendations or averaging the rank or rating predicted by the two methods. This is essentially what our naive hybrid approach does.

Soboroff et al. propose a novel approach to combining content and collaboration using latent semantic indexing (LSI) [20]. In their approach, first a *term-document matrix* is created, where each cell is a weight related to the frequency of occurrence of a term in a document. The term-document matrix is multiplied by the normalized ratings matrix to give a *content-profile matrix*. The singular value decomposition (SVD) of this matrix is computed. Using LSI, a rank-k approximation of the content-profile matrix is computed. Term vectors of the user's relevant documents are averaged to produce a centroid representing the user's profile. Now, new documents are ranked against each user's profile in the LSI space.

In Pazzani's approach [18], user profiles are represented by a set of weighted words derived from positive training examples using the Winnow algorithm. This collection of user profiles can be thought of as the *content-profile matrix*. Predictions are made by applying CF directly to the *content-profile matrix* (as opposed to the user-ratings matrix).

An alternate approach to providing content-based collaborative recommendations is used in Fab [3]. Fab uses relevance feedback to simultaneously mold a personal filter along with a communal "topic" filter. Documents are initially ranked by the topic filter and then sent to user's personal filters. A user then provides relevance feedback for that document, which is used to modify both the personal filter and the originating topic filter.

Basu et al. integrate content and collaboration in a framework in which they treat recommending as a classification task [4]. They use *Ripper*, an inductive logic program, to learn a function that takes a user and movie and predicts a label indicating whether the movie will be liked or disliked. They combine collaborative and content information, by creating features such as *comedies liked by user* and *users who liked movies of genre X*.

Good et al. [10] use collaborative filtering along with a number of personalized information filtering agents. Predictions for a user were made by applying CF on the set of other users and the active user's personalized agents. Our method differs from this by also using CF on the personalized agents of the other users.

In recent work, Lee [13] treats the recommending task as the learning of a user's preference function that exploits item content as well as the ratings of similar users. They perform a study of several mixture models for this task.

In related work, Billsus and Pazzani [5] use singular value decomposition to directly tackle the *sparsity* problem. They use the SVD of the original user-ratings matrix to project user-ratings and rated items into a lower dimensional space. By doing this they eliminate the need for users to have co-rated items in order to be predictors for each other.

9. CONCLUSIONS AND FUTURE WORK

Incorporating content information into collaborative filtering can significantly improve predictions of a recommender system. In this paper, we have provided an effective way of achieving this. We have shown how Content-boosted Collaborative Filtering performs significantly better than a pure content-based predictor, collaborative filtering, or a naive hybrid of the two.

CBCF elegantly exploits content within a collaborative framework. It overcomes the disadvantages of both collaborative filtering and content-based methods, by bolstering CF with content and vice versa. Further, due to the nature of the approach, any improvements in collaborative filtering or content-based recommending can be easily exploited to build a more powerful system.

Although CBCF performs consistently better than pure CF, the difference in performance is not very large (4%). We are currently attempting to boost the performance of our system by using the methods described in Section 7. In future, we also plan to test if our approach performs better than the other approaches that combine content and collaboration outlined in Section 8.

Acknowledgments

We would like to thank the Compaq Computer Corporation for generously providing the EachMovie dataset used in this paper. We are grateful to Vishal Mishra for his web crawler and many useful discussions. We also thank Joydeep Ghosh and Inderjit Dhillon for their valuable advice during the course of this work. This research was supported by the National Science Foundation under grant IRI-9704943.

10. REFERENCES

[1] *EachMovie dataset.* http://research.compaq.com/SRC/eachmovie.

[2] *Internet Movie Database.* http://www.imdb.com.

[3] M. Balabanovic and Y. Shoham. Fab: Content-based, collaborative recommendation. *Communications of the Association of Computing Machinery*, 40(3):66–72, 1997.

[4] C. Basu, H. Hirsh, and W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pages 714–720, July 1998.

[5] D. Billsus and M. J. Pazzani. Learning collaborative information filters. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML-98)*, pages 46–54, Madison, WI, 1998. Morgan Kaufmann.

[6] P. Cotter and B. Smyth. PTV: Intelligent personalized tv guides. In *Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 957–964, 2000.

[7] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification.* Wiley, New York, November 2000.

[8] D. Fisher, K. Hildrum, J. Hong, M. Newman, M. Thomas, and R. Vuduc. Swami: A framework for collaborative filtering algorithm development and evaluation. In *SIGIR 2000*, July 2000. Short paper.

[9] D. Goldberg, D. Nichols, B. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the Association of Computing Machinery*, 35(12):61–70, 1992.

[10] N. Good, J. B. Schafer, J. A. Konstan, A. Borchers, B. Sarwar, J. Herlocker, and J. Riedl. Combining collaborative filtering with personal agents for better recommendations. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, pages 439–446, July 1999.

[11] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR '99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 230–237, 1999.

[12] R. Kohavi, B. Becker, and D. Sommerfield. Improving simple Bayes. In *Proceedings of the European Conference on Machine Learning*, 1997.

[13] W. S. Lee. Collaborative learning for recommender systems. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML-2001)*, 2001.

[14] A. K. McCallum and K. Nigam. A comparison of event models for naive Bayes text classification. In *Papers from the AAAI 1998 Workshop on Text Categorization*, pages 41–48, Madison, WI, 1998.

[15] T. Mitchell. *Machine Learning.* McGraw-Hill, New York, NY, 1997.

[16] R. J. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the SIGIR-99 Workshop on Recommender Systems: Algorithms and Evaluation*, Bekeley, CA, 1999.

[17] R. J. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *"Proceedings of the Fifth ACM Conference on Digital Libraries"*, pages 195–204, San Antonio, TX, June 2000.

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

[19] P. Resnick, N. Iacovou, M. Sushak, P. Bergstrom, and J. Reidl. GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 Computer Supported Cooperative Work Conference*, New York, 1994. ACM.

[20] I. Soboroff and C. Nicholas. Combining content and collaboration in text filtering. In T. Joachims, editor, *Proceedings of the IJCAI'99 Workshop on Machine Learning in Information Filtering*, pages 86–91, 1999.

APPENDIX

The performance of the content-based predictor was evaluated using 10-fold cross-validation, in which each data set is randomly split into 10 equal-size segments and results are averaged over 10 trials. For each trial, one segment is set aside for testing, while the remaining data is available for training. To test performance on varying amounts of training data, a learning curve was generated by testing the system after training on increasing subsets of the overall training data. We generated learning curves for 132 users who had rated more than 200 items. The points on the 132 learning curves were averaged to give the learning curve in Figure 2.

Beyond Algorithms: An HCI Perspective on Recommender Systems

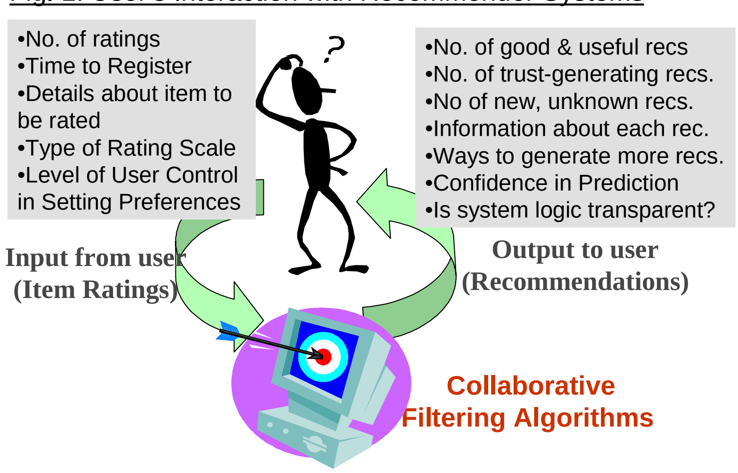
Kirsten Swearingen & Rashmi Sinha
SIMS, UC Berkeley, 94720
{kirstens, sinha}@sims.berkeley.edu

Abstract: The accuracy of recommendations made by an online Recommender System (RS) is mostly dependent on the underlying collaborative filtering algorithm. However, the ultimate effectiveness of an RS is dependent on factors that go beyond the quality of the algorithm. The goal of an RS is to introduce users to items that might interest them, and convince users to sample those items. What design elements of an RS enable the system to achieve this goal? To answer this question, we examined the quality of recommendations from and the usability of three book RS (Amazon.com, RatingZone & Sleeper) and three movie RS (Amazon.com, MovieCritic, Reel.com). Our findings indicate that from a user's perspective, an effective recommender system inspires trust in the system; has system logic that is at least somewhat transparent; points users towards new, not-yet-experienced items; provides details about recommended items, including pictures and community ratings; and finally, provides ways to refine recommendations by including or excluding particular genres. Users expressed willingness to provide more input to the system in return for more effective recommendations.

INTRODUCTION

A common way for people to decide what books to read or movies to watch is to ask their friends for recommendations. Online Recommender Systems (RS) attempt to create a technological proxy for this social filtering process. Previous studies of RS have mostly focused on the collaborative filtering algorithms that drive the recommendations (Delgado 2000, Herlocker 2000, Soboroff 1999). We conducted an empirical study to examine user's interactions with several online book and movie RS from an HCI perspective. We had two specific goals. Our first goal was to examine users' interaction with RS (i.e., input to the system, output from the system, and other interface factors) in order to isolate design features that go into the making of an effective RS. Our second goal was to compare, from the user's perspective, two ways of receiving recommendations: (a) from online RS and (b) from friends (the social recommendation process).

Fig. 1: User's Interaction with Recommender Systems



To achieve our first project goal, we did an empirical study of three book RS (Amazon.com, RatingZone's QuickPicks, and Sleeper) and three movie RS (Amazon.com, Moviecritic, and Reel.com). We chose this variety of online RS based on differences in interfaces (layout, navigation, color, graphics, and user instructions), types of input required, and information displayed with recommendations (see Appendix for the RS comparison chart). An RS may take input from users implicitly or explicitly, or a combination of the two (Schafer et. al 1999). Our study examined systems that relied upon explicit input.

The second goal of the study was to compare the performance of online RS to that of human recommenders—the friends of our test subjects. Results showed that the users' friends consistently provided better recommendations, i.e., a higher percentage of "good" and "useful" recommendations as compared to online RS (see Fig. 1). However, further analysis and post-test interviews revealed that users did find value in the online RS. (For a detailed discussion of the RS vs. friends methodology and findings, see Sinha & Swearingen, 2001.)

METHODOLOGY

Participants: A total of 19 people participated in our experiment. Each participant tested either 3 book or 3 movie systems, as well as evaluating recommendations made by 3 friends. Study participants were mostly students at the University of California, Berkeley. *Age range*: 20 to 35 years. *Gender ratio*: 6 males and 13 females. *Technical background*: 9 worked in or were students in technology-related fields, the other 10 were studying or working in non-technical fields.

Procedure: This study was completed during November 2000 – January 2001. For each of the three book/movie recommendation systems (presented in a random order), users completed the following tasks: (a) Completed online registration process (if any) using a false e-mail address so that any existing buying/browsing history would not color the recommendations provided during the experiment. (b) Rated items on each RS in order to get recommendations. (Some systems required users to complete a second step, where they were asked for more ratings to refine recommendations.) (c) Reviewed list of recommendations. (d) If the initial set of recommendations did not provide anything that was both new and interesting, users were asked to look at additional items. They were to stop looking when they found at least one book/movie they were willing to try, or they grew tired of searching. (e) Completed satisfaction and usability questionnaire for each RS. After the user had tested and evaluated all three systems, we conducted a post-test interview.

Independent Variables: (a) Item domain: books or movies (b) Source of recommendations: friend or online RS (c) Recommender System itself.

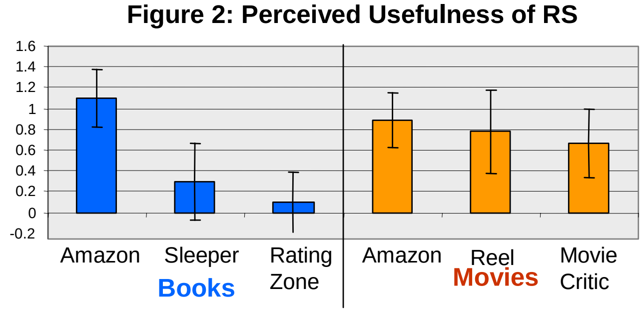
Dependent Measures:
(a) Quality of recommendations was evaluated using 3 metrics.
- *Good Recommendations:* Percentage of recommended items that the user liked. Good Recommendations were divided into the following two subcategories.
- *Useful Recommendations* were "good" recommendations that the user had not experienced before. This is the sum total of useful information for the user—ideas for new books to read / movies to watch.
- *Previously Liked Recommendations (Trust-Generating Recommendations)* were "good" recommendations that the user had already experienced and enjoyed. These are not "useful" in the traditional sense, but our study showed that such items indexed users' confidence in the RS.

(b) Overall satisfaction with recommendations and with RS.
(c) Time measures – time spent registering and receiving recommendations from the system

GENERAL DISCUSSION

a) **Users Perceived RS as being Useful:** Overall, users expressed a high level of overall satisfaction with online RS. Their qualitative responses in the post-test questionnaire indicated that they found the RS useful and intended to use the systems again.

Figure 2: Perceived Usefulness of RS



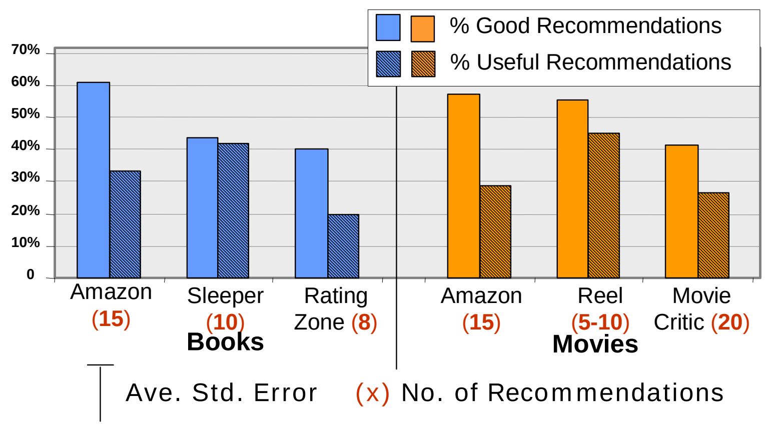
b) **Users did not Like All RS Equally:** However, not all RS performed equally well. As Figure 2 shows, though most systems were judged at least somewhat useful, Amazon Books was judged the most useful, RatingZone was judged not useful, while Sleeper was judged only moderately useful. This corresponds to the results of the post-test interviews, in which, of the 11 users who said they preferred one of the online systems, 6 named Amazon as the best (3 for Amazon-books and 3 for Amazon-movies), 3 preferred Sleeper, and 3 liked MovieCritic.

c) **What Factors Predicted Perceived Usefulness of System:** What factors contributed to the perceived usefulness of a system? To examine this question, we computed correlations between Perceived Usefulness and other aspects of a Recommender System (see Table 1). We found that certain elements correlated strongly with perceived usefulness, while others showed a very low correlation.

Table 1. Predicting Perceived Usefulness	
Factors that predict Perceived Usefulness	
Number of Good Recs.	0.53 **
Number of Useful Recs.	0.41 **
Detail in item description	0.35 **
Know reason for recs.? (Transparency)	0.31 *
Trust-Generating Recs.	0.30 *
Factors that don't predict RS Usefulness	
Time to get Receive Recs.	0.09
Number of Recs.	-0.02

As Table 1 shows, Perceived Usefulness correlated most highly with % Good and % Useful Recommendations. % Good Recommendations is indicative of the accuracy of the algorithm, and it is not surprising that it plays an important role in determining Perceived Usefulness of System. However, these two metrics (Good and Useful Recommendations) do not tell the whole story. For example, RatingZone's performance was comparable to Amazon and Sleeper, in terms of Good and Useful recommendations and yet it was neither named as a favorite nor deemed "Very Useful" by subjects. On the other hand, MovieCritic's performance was poor relative to Amazon and Reel, but several users named it as a favorite. Clearly, other factors influenced the users' perception of RS usefulness. Our next task was to attempt to isolate those factors.

Figure 3: "Good" & "Useful" Recommendations



Design Suggestions for Recommender Systems

To identify system elements that contributed to perceived overall effectiveness of a system, we analyzed both quantitative and qualitative data (post-test interviews, user comments and observations during test). Drawing on our analysis, we have attempted to offer design suggestions for Recommender Systems. These design suggestions are based on our interpretation of the qualitative and quantitative data we gathered during our study. To the degree possible, we have included figures, tables, user comments, and our own observations to support our reasoning. For some system elements, we do not have any specific recommendations (since our data did not allow any strong inferences). In such cases, we have attempted to define a range of design options, and the factors to consider in choosing a particular option. For purposes of discussion, we have divided our design suggestions into two broad categories: system input elements and system output elements.

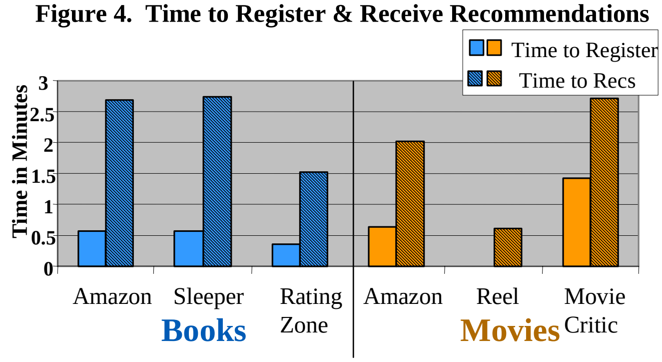
I) Design Suggestions: System Input Elements

I-a) Number of Ratings Required to Receive Recommendations / Time to Register

Our results indicate that a moderate increase in the number of ratings required does not have a strong negative impact on ease of use (see Table 1, above). Some of the systems that required the user to make many ratings (e.g. Amazon, Sleeper) were rated highly on satisfaction and perceived usefulness. Ultimately what mattered to users was whether they got what they came for: useful recommendations. Users appeared to be willing to invest a little more time and effort if that outcome seemed likely. They did express some impatience with systems that required a large number of ratings, e.g., with MovieCritic, which required users to rate 12 movies, and Rating Zone, which asked users to look at 50 items. However, the users' impatience seemed to have less to do with the absolute number of ratings and more to do with the way the information was displayed (e.g., only 10 movies on each screen, no detailed information or cover image with the title, necessitating numerous clicks in order to rate each item). For more details on presentation of rating information and interface issues, see sections I-b and II-e, below.

We recorded the time taken by users to register on the site, and to complete all the steps necessary to receive recommendations. These time measures did not seem to directly affect the perceived usefulness of the system, as seen in Table 1, above. As Figure 3 shows, the systems that allowed users to receive recommendations the most quickly were not the ones that provided the most useful suggestions.

Figure 4. Time to Register & Receive Recommendations



We had also asked users if they thought any system asked for too much personal information during the registration process. Most systems required users to indicate information such as name, e-mail address, age, and gender. The users did not mind providing this information and it did not take them a long time to do so.

- *"… there wasn't a lot of variation in the results… I'd be willing to do more rating for a wider selection of books." (Comment about Amazon)*
- ***"****There could be a few (2 or 3) more questions to gain a clearer idea of my interests…maybe if I like historical novels, etc.?"(Comment about RatingZone)*

Design Suggestion: Designers of recommendation systems are often faced with a choice between enhancing ease of use (by asking users to rate fewer items) or enhancing the accuracy of the algorithms (by asking users to provide more ratings). Our suggestion is that it is fine to ask to the users for a few more ratings if that leads to substantial increases in accuracy.

I-b) Information about Item Being Rated

The systems differed in the amount of information they provided with the item to be rated. Some, such as RatingZone (version 1), provided only the title. If a user was not sure whether he/she had read the item, there was no way to find out more information to jog his/her memory. Other systems, such as MovieCritic, Amazon and RatingZone (version 2), provided additional information but located it at least one click away from the list of items to be rated. Finally, systems such as Sleeper provided a full plot synopsis along with the cover image. Sleeper differed from the other RS in another important way. Rather than trying to develop a gauge set of popular items that people would be likely to have read or seen, Sleeper circumvented the problem by selecting a gauge set of obscure items, then asking "how interested are you in books like this one?" instead of "what did you think of this book?" This meant that users were empowered to rate every item presented, instead of having to page through long lists, hoping to find rate-able items.

- *9 of the 15 she hadn't heard of—"I have to click through to find out more info." (Sighing.) "Lots of clicking!"(Comment about Amazon)*
- *Worried because she hadn't read many of the books [to be rated].(Comments about RatingZone)*
- *"I don't read too many books--brief descriptions were helpful" (Comment about Sleeper)*

Design Suggestion: Satisfaction and ease-of-use ratings were higher for the systems that collocated some basic information on the rating page. Cover image and plot synopses received the most positive comments, but future studies could identify other crucial elements for inclusion.

I-c) Rating Scales for Input Items

Figure 5. Sleeper Rating Scale

The RS used different kinds of rating scales for input ratings. MovieCritic used a 9-point Likert Scale, Amazon asked users for a favorite author / director, while Sleeper used a continuous rating bar. Some users commented favorably on the continuous rating bar used by Sleeper (See Figure 4), which allowed them to express gradations of interest level. Part of the reaction seemed to be to the novelty of the rating method. The only negative comments on rating methods were regarding Amazon's open text-box for "Favorite item." "Three of the users did not want to select a single item (artist, author, movie, hobby) as "favorite;" one user tried to enter more than one item in the "Favorite Movie" textbox, only to receive an error.

- *"I liked rating using the shading"(Comment about Sleeper's rating scale)*
- *"Interesting approach, [it was] easy to use."(Comment about Sleeper's rating scale).*

Design Suggestion: We do not have design suggestions in this area, but recommend pre-testing the rating scale with users; we also think that user's preference for continuous scale vs. discrete scales should be studied further.

I-e) Filtering by Genre

MovieCritic provided examples of both effective and ineffective ways to give users control over the items that are recommended to them. The system allowed users to set a variety of filters. Almost all of the users commented favorably on the genre filter—they liked being able to quickly set the "include" and "exclude" options on a list of about 20 genres. However, on the same screen, MovieCritic offered a number of advanced features, such as "rating method" and "sampling method" which were confusing to most users. Because no explanation of these terms was readily available, users left the features set to their default values. Although this did not directly interfere with the recommendation process, it may have negatively affected the sense of control which the genre filters had so nicely established.

- *"Good they show how to update—I like this."(Comment about MovieCritic)*
- *"Amazon should have include/exclude genre, like MovieCritic" (Comment about Amazon & MovieCritic)*
- *"No idea what a rating method or sampling method are [in Preferences]"(Comment about MovieCritic)*

Design Suggestion: Our design suggestion is to include filter-like controls over genres, but to make them as simple and self-explanatory as possible.

II) Design Suggestions: System Output Elements

II-a) Accuracy of Algorithm

As discussed earlier, Perceived Usefulness of systems correlated highly with % Good and % Useful recommendations. Both our qualitative and quantitative data give support for the fact that accurate recommendations are the backbone of an effective RS. The design suggestions that we are discussing are useful only if the system can provide accurate recommendations.

II-b) Good Recommendations that have been Previously Experienced (Trust-Generating Recommendations)

As Table 1 shows, Good Recommendations with which the user has previously had a positive experience correlate with Perceived Usability of systems. Such recommendations are not useful in the traditional sense (since they do not offer any new information to the user), but they index the degree of confidence a user can feel in the system. If a system recommends a lot of "old" items that the user has liked previously, chances are, the user will also like "new" recommended items.

Fig. 6: Perceived Usefulness of System as a Function of Trust-Generating Recommendations

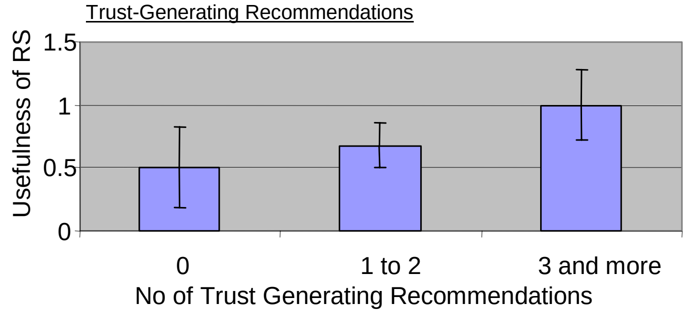


Figure 6 shows that the perceived usefulness of a recommender system went up with an increase in the number of trust-generating recommendations.

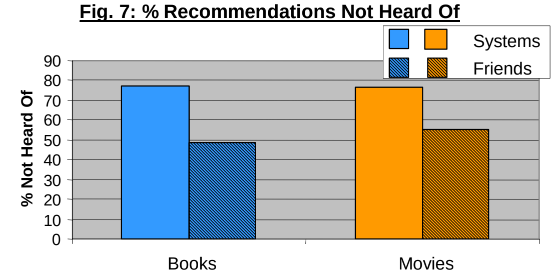
- *"I made my decision because I saw the movie listed in the context of other good movies" (Comment about Reel)*

Design Suggestion: Our design suggestion is that systems should take measures to enhance user's trust. However, it would be difficult for any system to insure that some percentage of recommendations were previously experienced. A possible way to facilitate this would be to generate some very popular recommendations, classics that the user is likely to have watched / read before. Such items might be flagged by a special label of some kind (e.g., "Best Bets").

II-c) Recommendations of New, Unexpected Items

Again, this concern has less to do with design and more to do with the algorithm driving the recommendations. It complements the previous point regarding trust-generating items. Five of our users stated that their favorite RS succeeded by expanding their horizons, suggesting items they would not have encountered otherwise.

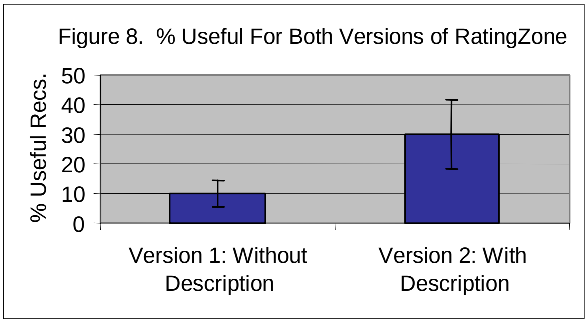
Fig. 7: % Recommendations Not Heard Of



- *"A number of things I hadn't heard of. Some guesses were more out there than friends, but[it was] nice to be surprised….90% of friends' books I'll want to read, but I already knew I wanted to read these. I want to be stretched, stimulated with new ideas."(Comment about Amazon)*
- *"Sleeper suggested books I hadn't heard of. It was like going to Cody's [a local bookstore]—looking at that table up front for new and interesting books." (Comment about Sleeper)*

Design Suggestion: To achieve this design goal, RS could include recommendations of new, just released items. Also RS could recommend a few lesser-known items.

II-d) Information about Recommended Items



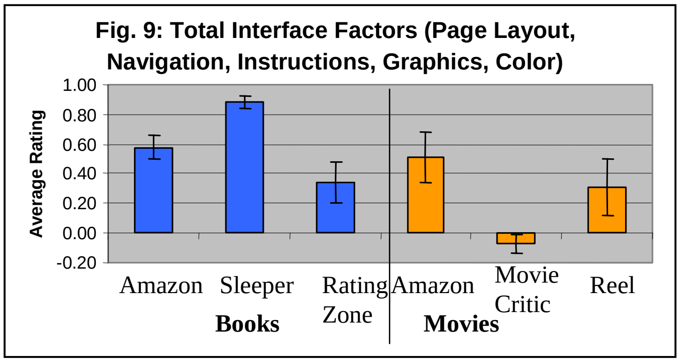
As with the rating input process, the presence of longer descriptions of individual items correlated positively with both the perceived usefulness and ease of use of RS. This indicates that users like to have more information about the recommended item (book / movie description, author / actor / director, plot summary, genre information, reviews by other users). Reviews and ratings by other users seemed to be especially important. Several users indicated that reviews by other users helped them in their decision-making. Similarly, people commented that pictures of the item recommended were very helpful in decision-making. Cover images often helped users recall previous experiences with the item (e.g., they had seen that movie in the video store, read a review of the book etc.).

This finding was reinforced by the difference between the two versions of Rating Zone. The first version of RatingZone's Quick Picks did not provide enough information and user evaluations were almost wholly negative as a result. A different problem occurred at MovieCritic, where detailed information was offered but users had trouble finding it, due to poor navigation design.

- *"Of limited use, because no description of the books."(Comment about RatingZone, Version 1)*
- *"Red dots [Predicted ratings] don't tell me anything. I want to know what the movie's about."(Comment about MovieCritic)*
- *"I liked seeing cover of box in initial list of result… The image helps."(Comment about Amazon)*

Design Suggestion: We recommend providing clear paths to detailed item information, and offering some kind of a community forum for users to post comments as a relatively easy way to dramatically increase the efficacy of the system.

II-e) Interface Issues



From the user's point of view, interface matters, mostly when it gets in the way. Navigation and layout seemed to be the most important factors--they correlated with ease of use and perceived usefulness of system, and generated the most comments, both favorable and unfavorable. For example, MovieCritic was rated negatively on layout and navigation. In general MovieCritic performed well in terms of Good and Useful recommendations. Users' comments indicated that the navigation problems with MovieCritic might have lead to its low overall rating. Users did not have strong feelings about color or graphics and these items did not correlate strongly with perceived usefulness.

- *"Don't like how recommendations are presented. No information easily accessible. Not clear how to get info about the movie. Didn't like having to use the Back button [to get back from movie info]"(Comment about MovieCritic)*
- *"Didn't like MovieCritic--too hard to get to descriptions."(Comment about MovieCritic)*

Design Suggestion: Our design suggestion is to invest time in user-testing the navigational structure of the RS, as deficiencies in this area can impact user satisfaction.

II-f) Predicting the Degree of Liking for Recommended Items

Some systems do not just recommend items users might like, they also predict the degree to which the person will like the item. Within our sample of systems, only Sleeper and MovieCritic provided such predictions (Amazon has recently added such a rating to its recommendation engine).

Users seemed to be mostly neutral about the "degree of liking" predictions; they did not help or hinder users' interactions with the system. However, such ratings can make users more critical of the recommendations. It would be easy for a user to lose confidence in a system that predicted a high degree of liking for an item he/she hates. Another potential problem is if the system recommends items with low or medium "predicted liking" ratings. In such cases (as with Sleeper) users were confused about why the system recommended such items—the sparsity of items in the database was not visible, so users were left feeling like "hard to please" customers, and feeling unsure about whether to seek out the items given such tepid endorsements by the RS. Degree of liking may also be expressed categorically (as with MovieCritic). MovieCritic divided items into Best Bets or Worst Bets and some users liked this approach.

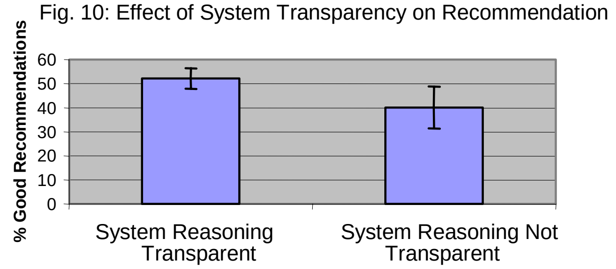
- *"All recommendations were in the middle of the Interested/Not Interested scale."(Comment about Sleeper)*
- *"So, so [in terms of usefulness]. Many books it recommended were ones I would be very interested in, yet they thought otherwise."(Comment about Sleeper)*

Design Suggestion: Our design suggestion is that presenting the degree of liking is a high-risk feature. A system would need to have a very high degree of accuracy for users to benefit from this feature. Degree of liking is useful information for the system itself, in that it can be used to sort the recommendations.

II-g) Effect of System Transparency

Users liked to understand what was driving a system's recommendations. Figure 10 shows that % Good Recommendations was positively related to Perceived System Transparency. This effect also surfaced in the comments made by users.
On the other hand, some users, particularly those with a technical background, were irritated when a system's algorithm seemed too simplistic: "Oh, this is another Oprah book," or "These are all books by the author I put in as a Favorite."

Fig. 10: Effect of System Transparency on Recommendation



- *"I really liked the system, but did not understand the recommendations." (Comment about Sleeper)*
- *"Don't know why computer books were included in refinement step. Didn't like any of them." (Comment about Amazon)*
- *"This movie was recommended because Billy Bob Thornton is in it. That's not enough."(Comment about MovieCritic)*
- *"They only recommended books by the author I picked. Lazy!"(Comment about Amazon)*

Design Suggestion: Users like the reasoning of RS to be at least somewhat transparent. They are confused if all recommendations are unrelated to the items they rated. RS should try to recommend at least some items that are close to (i.e., by the same or similar author , or in the same style as) the rated items.

We also noticed that users are critical if the system logic seems too simplistic. This might be due to a mismatch between user expectations and system capabilities. Our design suggestion is to communicate clearly the primary purpose of the RS, so as to manage the expectations of those who invest the time to use it. Communicating the reason a specific item is recommended also seems to be good practice—unfortunately, Amazon added this capacity after our study was completed so we were unable to gather feedback on its perceived utility. RS could offer different kinds of recommendations, such as "More by same author" "More from same genre." This would give users a choice in the kind of recommendations they receive.

The Recipe for an Effective Recommender System: Different Strokes for Different Folks
Comments such as the ones above these led us to an important realization, explored during post-test interviews: the "goodness" of a recommendation and the perceived usefulness of an RS depends heavily upon the user's expectations. Even within our small group of users, we discovered a wide range of recommendations needs. Below, we offer a tentative categorization of user needs from RS.

- **Reminder recommendations,** mostly from within genre ("I was planning to read this anyway, it's my typical kind of item")
- **"More like this" recommendations,** from within genre, similar to a particular item ("I am in the mood for a movie similar to GoodFellas")
- **New items,** within a particular genre, just released, that they / their friends do not know about
- **"Broaden my horizon"** recommendations (might be from other genres)

LIMITATIONS OF PRESENT STUDY

Conclusions drawn from this study are somewhat limited by several factors. (a) One limitation of our experiment design was that we handicapped the systems' collaborative filtering mechanisms by requiring users to simulate a first-time visit, without any browsing, clicking, or purchasing history. This deprived systems such as Amazon and MovieCritic of a major source of strength--the opportunity to learn user preferences by accumulating information from different sources over time. (b) A second limitation is that we did not study a random sample of online RS. As such, our results are limited to the systems we chose to study. (c) Finally, this study suffers from the same limitations as any other laboratory study: we do not know if users will behave in the same way in real life as in the lab.

REFERENCES

- Joaquin Delgado. "Agent-Based Information Filtering and Recommender Systems." Ph.D thesis. March 2000.
- David Goldberg, Daniel Nichols, Brian M. Oki, and Douglas Terry. Using Collaborative Filtering to Weave an Information Tapestry." *Communications of the ACM*, December 1992. 32 (12)
- Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins, "Eigentaste, A Constant-Time Collaborative Filtering Algorithm," *Information Retrieval*, 4(2), July 2001.
- Jonathan L. Herlocker, Joseph A. Konstan, John Riedl. "Explaining collaborative filtering recommendations." In *Proceeding on the ACM 2000 Conference on computer supported cooperative work*, 2000, Pages 241 – 250
- Don Peppers and Martha Rogers, Ph.D. "I Know What You Read Last Summer," Inside 1to1. Oct. 21, 1999. http://www.1to1.com/articles/il-102199/index.html
- P. Resnick and H.R. Varian, "Recommender systems." *Communications of the ACM*, 1997. 40(3) 56-58.

- Rashmi Sinha and Kirsten Swearingen. "Benchmarking Recommender Systems." *Proceedings from DELOS workshop on personalization and recommender systems*, June 2001
- Ian M. Soboroff and Charles K. Nicholas "Combining Content and Collaboration in Text Filtering," *Proceedings of the IJCAI 99 Workshop on Machine Learning and Information Filtering*, Stockholm, Sweden, August 1999.
- Shawn Tseng and B. J. Fogg, "Credibility and Computing Technology," *Communications of the ACM*, special issue on Persuasive Technologies, 42 (5), May 1999.

APPENDIX: Description of Recommender Systems Examined in Study

Note: This study was completed during November 2000 – January 2001. Since then, 3 of the RS sites (Amazon, RatingZone, and MovieCritic) have altered their interfaces to various degrees.

	Description of Recommendation System				
User Input Aspect	**Amazon (both books and movies)**	**Sleeper**	**RatingZone**	**Reel**	**MovieCritic**
How many items must a user rate to receive recommendations?	1 favorite item in each of 4 different categories, 16 more items in refinement step	15 items to rate (mandatory)	50 items to review, all optional to rate	1 item at a time	12 items to rate (mandatory)
Who generates items to rate?	User, initially.	System	System	User	System or user
Demographic information required	Name, e-mail address, age	Name, e-mail address	Name, e-mail address, age, gender, and zip	Nothing	Name, e-mail address, gender, age
Item rating scale	Favorite, then checkbox for "recommend items like this"	Shaded bar (range from "interested" to "not interested)	Checkbox for "I liked it"	No rating, just enter the movie you want matched	11 point scale ("Loved it" to "Hated it" to "Won't see it")
Users could specify interest in particular item type or genre	No	No	Yes	No	Yes
System Rec. Aspects	**Amazon**	**Sleeper**	**RatingZone**	**Reel**	**MovieCritic**
Item information (titles only, cover images, synopsis etc.)	Title, cover image, synopsis	Title, cover image, synopsis	RZ Version 1: Title, # of pages, year of pub. RZ Version 2: added link to Amazon.	Title, cover image, brief description,	Screen 1: title. Screen 2: predicted ratings and other ratings Screen 3: IMDB
Information about system's confidence in recommendation	No	Yes	No	No	Yes
Information on other users' ratings	Yes	No	No	No	Yes

Short-Length Presentations

Information access in Implicit Culture framework

E. Blanzieri[1,3], P. Giorgini[2], P. Massa[1], and S. Recla[1]

[1] ITC-irst Trento - Italy - {blanzier,massa,recla}@itc.it
[2] Dept. of Mathematics - University of Trento - Italy - pgiorgini@science.unitn.it
[3] Present: Dep. of Psychology - University of Turin - Italy - blanzier@psych.unito.it

Abstract. The goal of a System for Implicit Culture Support (SICS) is to establish an implicit culture phenomenon, namely when the elements of a set behave according to the culture of a generally different group of agents.. Earlier work claimd that Implicit culture support can be seen as a generalization of Collaborative Filtering and it can be useful for both artificial and human agents. In this paper, we recall the concept of Implicit Culture, show how it is useful for automatically exploit tacit knowledge and we present an implementation of a System for Implicit Culture Support. Finally, we present the application of the SICS to the eCulture Brokering System, a multi-agent system aimed to mediate the access to cultural information.

1 Introduction

Systems for Implicit Culture Support (SICSs in the following) [2] have the goal of estabilishing an Implicit Culture phenomenon. Informally, Implicit Culture is the relation existing between a set and a group of agents such that the elements of the set behave according to the culture of the group. Supporting Implicit Culture is effective in solving the problem of improving the performances of agents acting in an environment where more-skilled agents are active. This concept can help the analysis of existing systems and suggest ideas for the synthesis of new ones. In fact, support of Implicit Culture can be useful for various applications: Computer Supported Collaborative Work, Group profiling, Cognitive modelling of social phenomena, e-books, Computer Mediated Communication Systems and, as we briefly present in the following, generalization of collaborative filtering, Knowledge Managment and requirements and interaction control of agent-based systems.

Collaborative filtering [9, 19, 12] is a popular technique exploited in recommendation systems. The goal is information filtering, namely to extract from a usually long list of items, e.g. links or products, a little set that the user could prefer. Collaborative filtering exploits correlations in the pattern of preferences expressed actively o passively by other users in terms of ratings. Differently from the content-based filtering, collaborative filtering does not rely on the content or shape of objects. The central idea is to automate the process of recommending items to a user on the base of the opinions of people with similar preferences.

Billsus and Pazzani have tackled collaborative filtering problems in a Machine Learning perspective [1]. As suggested by Blanzieri and Giorgini [2] collaborative filtering can be seen as a System for Implicit Culture Support (SICS).

In Knowledge Management, generally knowledge is categorized as being either codified (explicit) or tacit (implicit). Knowledge is said being explicit when it is possible to describe and share it among people through documents and/or information bases. Knowledge is said being implicit when it is embodied in the capabilities and abilities of the members of a group of people. In [15], knowledge creation processes have been characterized in terms of tacit and explicit knowledge transformation processes, in which, instead of considering new knowledge as something that is added to the previous, they conceive it as something that transforms it. Implicit Culture can be applied successfully in this context. In particular, the idea is to build systems able to capture implicit knowledge, but instead of sharing it among people, change the environment in order to make new people behave in accordance with this knowledge.

Advantages of Implicit Culture support has been proposed also for artificial agents [7, 13], in particular for controlling the requirements of agent-based systems and supporting their interaction. Autonomy of agents, unknown properties of the environment and insertion of new agents do not allow to foresee completely the multi-agent system's behavior in the modeling phase. As a consequence, the overall system can fail to fulfill the desired requirements. In particular, requirements should persist after a composition changing of the group of agents, that is the new agents should act consistently with the culture of the group. Using SICSs, it is possible to modify the view that the agents have of the environment and, consequently, change the set of possible actions that the agents can perform in the environment. Working on the possible actions, a SICS is able to lead the new agents to act consistently with the behavior of the group.

The architecture of SICS proposed in [2, 7] relies on the exploitation of learning techniques. In particular it is possible to identify two learning problems: (i) induction of a cultural theory on the behavior patterns of the group and (ii) prediction of a scene such that the elements of the set will behave consistently with the cultural theory. The first problem can be solved by standard data mining techniques. In this paper we present an implemented SICS that solves the second problem exploiting an original generalization of a memory-based Collaborative Filtering algorithm. The SICS is applied to two different information access problems.

The paper is organized as follows. Section 2 presents the SICS architecture and the learning problems. In Section 3 we show a solution to the problem of prediction of the scene and in Sections 4 and 5 we present some experimental results and a real-world application, respectively. The final section hosts conclusions and future work.

2 Systems for Implicit Culture Support

The goal of a SICS is to establish an implicit culture phenomenon. In the following, we informally introduce the notions of implicit culture and implicit culture phenomenon (Appendix A reports the formal definitions given in [13]). The second part of the section presents the general architecture of a SICS and shows how it relies on learning techniques.

An Implicit Culture phenomenon is a pair composed by a set and a group of agents such that the elements of the set behave according to the culture of the group and Implicit Culture is the relation between the elements of the pair. The definitions are expressed in terms of expected situated actions and cultural constraint theories. Some assumptions underlie these concepts.

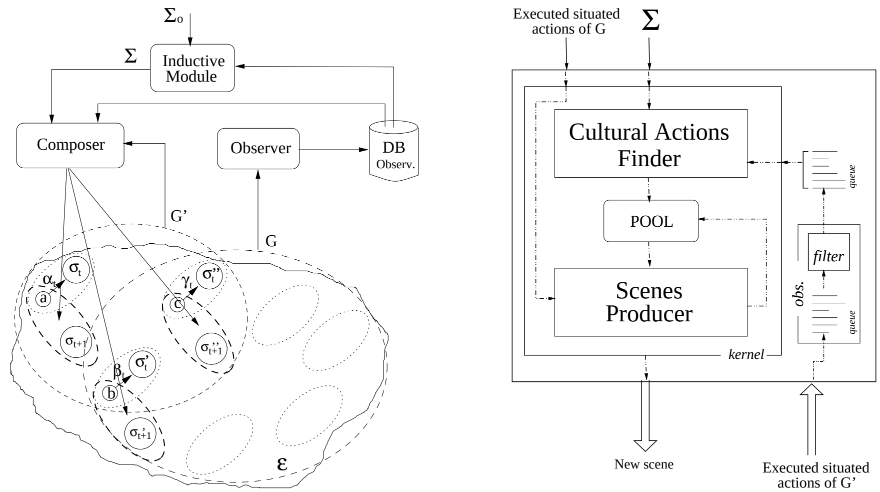
We assume that the agents perform situated actions. Agents perceive and act in an environment composed of objects and other agents. In this perspective, agents are objects that are able to perceive, act and, as a consequence of perception, know. Before executing an action, an agent faces a scene formed by a part of an environment composed of objects and agents. Hence, an agent executes an action in a given situation, namely the agent and the scene at a given time. After a situated action has been executed, the agent faces a new scene. At a given time the new scene depends on the environment and on the situated executed actions.

Another assumption states that the expected situated actions of the agents can be described by a cultural constraint theory. The action that an agent executes depends on its private states and, in general, it is not deteministically predictable with the information available externally. Rather, we assume that it can be characterized in terms of probability and expectations. Given a group of agents we suppose that there exists a theory about their expected situated actions. Such a theory can capture knowledge and skills of the agents about the environment and so it can be considered a cultural constraint of the group.

We call Implicit Culture a relation between a set of agents G' and a group G such that the agents of G' perform actions that satisfy a cultural constraint for G. When a set and a group of agents are in Implicit Culture relation, we have an Implicit Culture phenomenon. The definitions do not require the empirical validation of the cultural constraint theory against the executed actions of G.

The definition of Implicit Culture (Appendix A briefly discusses the use of "implicit") does not give sufficient conditions for its realization, posing the problem of its support in practise. The general architecture of a SICS proposed in [2] allows to achieve the goal of estabilishing an implicit culture phenomenon following two steps. First, the elaboration of a cultural constraint theory $\mathbf{\Sigma}$ from a given domain and a set of situated executed actions of a group G. Second, the proposal to a group G' of a set of scenes such that the expected situated actions of the set of agents G' satisfies $\mathbf{\Sigma}$. Both the steps present learning problems.

A general SICS (see Figure 1-a) consists of three components: observer, inductive module and composer. The observer stores the situated executed actions of a group of agents G in oder to make them available for the other components. The inductive module uses these actions to produce a cultural constraint theory



(a) Architecture (b) The composer in detail

Fig. 1.

$\mathbf{\Sigma}$ for G. Finally, the composer, using the theory $\mathbf{\Sigma}$ and the actions, manipulates the scenes faced by a set of agents G' in such a way that their expected situated actions are cultural action w.r.t G. As a result, the agents of G' executes (on average) cultural actions w.r.t G (implicit culture phenomenon).

In Figure 1-a the composer proposes to the agents a, b, and c the scenes σ_{t+1}, σ'_{t+1}, and σ''_{t+1}, respectively. Notice that in this case the agents b and c belong to both G and G'. This means that also their situated actions are stored in DB and thus they are used to elaborate the theory $\mathbf{\Sigma}$ and the new scenes.

In general, our implemented architecture accepts cultural theories expressed by a set of rules of the form:

$$A_1 \wedge \cdots \wedge A_n \rightarrow C_1 \wedge \cdots \wedge C_m$$

in which $A_1 \wedge \cdots \wedge A_n$ is said antecedent and $C_1 \wedge \cdots \wedge C_m$ consequent. The idea is to express that "if in the past the antecedent has happened, then there exists in the future some scenes in which the consequent will happen". Antecedent and consequent are conjunctions of atoms, namely two types of predicates: observations on an agent and conditions on times. For instance, $\mathtt{request(x,y,s,t_1)}$ is a predicate of the first type, that says that the agent $\mathtt{x}$ requests the agent $\mathtt{y}$ for the service $\mathtt{s}$ at time $\mathtt{t_1}$; whereas $\mathtt{less(t_1,t_2)}$ is an example of the second type and it simply says that $\mathtt{t_1<t_2}$.

The composer proposes to a set of agents G' a set of scenes such that their expected situated actions satisfy a cultural constraint theory $\mathbf{\Sigma}$ for a group G. The main idea is splitting the problem in two sub-problems: (i) find the cultural actions and (ii) find the scenes where such actions are the expected situated actions. Figure 1-b shows the composer in detail. Basically, the composer consists of two main submodules and an additional component:

- the *Cultural Actions Finder* (CAF), that takes as inputs the theory $\mathbf{\Sigma}$ and the executed situated actions of G', and produces as output the cultural actions w.r.t. G (namely, the actions that satisfy $\mathbf{\Sigma}$). The CAF matches the executed situated actions of G' with the antecedents of the rules of $\mathbf{\Sigma}$. If it finds an action that satisfies the antecedent of a rule, then it takes the consequent of the rule as a cultural action.
- the *Scenes Producer* (SP), that takes one of the cultural action produced by the CAF and, using the executed situated actions of G, produces scenes such the expected situated action is the cultural action.
- the *Pool*, an additional component, which manages the cultural actions given as input from the satisfaction submodule. It stores, updates, and retrieves the cultural actions, and solves possible conflicts among them.

The SICS architecture requires the solution of two learning problems. A problem of induction of the cultural constraint theory (Inductive Module) and a problem of prediction of scenes (Scenes Producer).

Definition 1 (Inductive Module Problem). *Given a set of situated executed actions performed by the agents of G, find a cultural constraint theory.*

Definition 2 (Scene Producer Problem). *Given a set of situated executed actions of the agents of G and G', and given a cultural action α for the agent x, find a scene s such that the expected situated action of x in the scene s is α.*

The Inductive Module Problem is a rather standard learning problem: inducing the behavior patterns of a group and it is possible to solve using standard data mining techniques . As we previously noted, we do not require any specific validation of the theory. Obviously, very different effects will be reached depending on the fact that the theory is validated or not. In the Scene Producer Problem the request is on the effectiveness of the scene w.r.t the goal of producing the execution of a given action, namely its *persuasiveness*.

3 A Solution to the Scene Producer Problem

The solution exploits the principles of instance-based learning (namely, memory-based or lazy). Given a cultural action α for the agent x that performed actions on the set of scenes $S(x)$, the algorithm used in the SP consists of three steps:

1. find a set of agents Q that performed actions similar to α;
2. select a set of agents $Q' \subseteq Q$ similar to x and the set of scenes S in which they performed actions;
3. select and propose to x a scene of S.

Figure 2 shows the algorithm used in step 1. An agent y is added to the set Q if the similarity $sim(\beta_y, \alpha)$ between at least one of its situated executed actions β_y and α is greater than the minimum similarity threshold T_{min}. The scenes s in which the β_y actions have been executed are added to $S(y)$, that is the set of

```
for all y ∈ G'
    for all situated executed actions β_y of y
        if sim(β_y, α) > T_min then {
            if y ∉ Q then  y → Q
            s → S(y)
        }
```

Fig. 2. The algorithm for step 1

scenes in which y has performed actions similar to α. Sections 4 and 5 contain two examples of similarity function (eq. 5 and 7).

Step 2 selects in Q the k nearest neighbors to x with respect to the agent similarity defined as follows:

$$w_{x,y} = \frac{1}{|S_{xy}|} \sum_{s \in S_{xy}} \frac{1}{N_x(s) N_y(s)} \sum_{\beta_x \in N_x(s)} \sum_{\beta_y \in N_y(s)} sim(\beta_x, \beta_y) \tag{1}$$

where $S_{xy} = S(x) \cap S(y)$ is the set of scenes in which both x and y have executed at least an action. $N_x(s)$ and $N_y(s)$ are the set of actions that x and y have respectively performed in the scene s. Eq. 1 can be replaced by a domain-dependent agent similarity function if needed (e.g., Eq. 4 in Section 4).

Step 3 selects the scenes in which the cultural action is the expected situated action. To do this, firstly we estimate for any scene $s \in S = \bigcup_{y \in Q} S(y)$ the similarity value between expected action and cultural action, and then we select the scene with the maximum value. The function to be maximized is the expected value $E(sim(\beta_x, \alpha)|s)$, where β_x is the action performed by the agent x, α is the cultural action, and $s \in S$ is the scene in which β_x is situated. The following estimate is used:

$$\hat{E}\left(sim(\beta_x, \alpha)|s\right) = \frac{\sum_{u \in Q'} \hat{E}_1\left(sim(\beta_u, \alpha)|s\right) * w_{x,u}}{\sum_{u \in Q'} w_{x,u}} \tag{2}$$

that is we calculate the weighted average of the similarity of the expected actions for the neighbor of the scene, $w_{x,u}$ is the similarity between the agent x and the agent u, whereas E_1 is estimate as follows:

$$\hat{E}_1\left(sim(\beta_u, \alpha)|s\right) = \frac{1}{|N_u(s)|} \sum_{\beta_u \in N_u(s)} sim(\beta_u, \alpha) \tag{3}$$

that is the average of $sim(\beta_u, \alpha)$ over the set of actions $N_u(s)$ performed by u in s.

The algorithms described above and the general architecture described in Section 2 are fully implemented in Java using XML for expressing the cultural constraint theory.

4 Experiments in a recommendation-problem domain

Collaborative filtering can be seen as a particular SICS. In this section we present the results of some experiments aimed to compare collaborative filtering and the SICS parametrized for a recommendation problem. The goal of the experiments is validating the system against a well-estabilished method on a particular domain.

For collaborative filtering we use a memory and neighborhood based algorithm presented by Herlocker et al. in [10]. The algorithm consists of three basic steps. In the first step all the users are weighted with respect to their similarity with the active user by *Pearson correlation coefficient*:

$$w_{a,u} = \frac{\sum_{i=1}^{m} (r_{a,i} - \overline{r}_a) * (r_{u,i} - \overline{r}_u)}{\sigma_a * \sigma_u} \tag{4}$$

where m is the number of objects co-valutated, $r_{a,i}$ is the ranking given by the user a to the object i, $\overline{r}_a$ and σ_a are respectively the average and variance of the rankings of a. In the second step the best k_{CF} correlates are picked in order to compute a prediction (third step) using the *deviation-from-mean* approach introduced in [19].

The SICS used for the experiments is characterized by: a set of agents (users) $\mathcal{P} = \{u_1, \cdots, u_n\}$; a set of objects $\mathcal{O} = M \cup V$ (M is the set of items and $V = \{0, 0.2, \cdots, 0.8, 1\}$ the set of the possible votes) and a set of possible actions $\mathcal{A} = \{vote, request\}$. For a recommender system the cultural theory is specified in advance so no inductive module is needed:

$$\forall x \in P, m \in M: \quad request(x) \to vote(x, m, v_{max})$$

is expressed by the following rule:

$$request(x, t_1) \to vote(x, m, v_{max}, t_2) \wedge less(t_1, t_2)$$

that states that if x requests a suggestion, then x will assign the maximum vote to the proposed item. For a recommender system we require the satisfation of the user with the recommended items. For similarity between actions we use the following domain-dependent function:

$$sim(A_1, A_2) = \begin{cases} 0 & if\ (A_1 = vote \wedge A_2 = request) \vee \\ & \quad (A_1 = request \wedge A_2 = vote) \vee \\ & \quad (A_1 = vote(x, o, v_1) \wedge A_2 = vote(y, p, v_2) \wedge o \neq p) \\ 1 & if\ A_1 = A_2 = request \\ 1 - |v_1 - v_2| & if\ (A_1 = vote(x, o, v_1) \wedge A_2 = vote(y, p, v_2) \wedge o = p) \end{cases} \tag{5}$$

that means that the similarity is zero when the actions are different or when they are both *vote* but about distinct objects; it is maximum (namely 1) when the actions are both *request*; and finally, it is $1 - |v_1 - v_2|$ when the actions are both *vote* about the same object, but with different numerical votes.

For the experimentation we used the database EachMovie [14] that collects data of 72961 users who voted 1623 movies. We built a dataset considering the

action sim	eq. 5 (domain-dependent similarity)								eq 7
agent sim	eq. 4 (Pearson Correlation coefficient)							eq. 1	eq. 4
sim min	T_{min}=0						T_{min}=0.8	T_{min}=0	T_{min}=0
neighbors	k=10	k=30	k=50	k=80	k=150	k=300	k=80	k=80	k=80
ranks									
1	25.4	57.6	65.3	68.6	68.6	69.5	66.1	58.0	6.8
2	14.4	11.0	11.0	10.2	10.2	11.0	12.7	14.3	4.2
3	8.5	7.6	6.8	7.6	8.5	6.8	6.8	8.4	4.2
4	10.2	5.1	5.1	4.2	4.2	3.4	5.9	5.9	4.2
5	6.0	2.5	1.7	0.0	0.0	1.7	0.0	1.7	5.1
over 5	35.5	16.2	10.1	9.4	8.5	7.6	8.5	11.7	75.5

Table 1. Experimental results. Distribution in percentage of the items proposed by SICS in the rankings position proposed by the CF algorithm with $K_{CF} = 30$.

first 50 movies and the 119 users with at least one vote among the first 300 and we run a leave-on-out w.r.t the users. The CF algorithms returns a list of estimated ranking while in this case the SICS returns only a scene composed by a movie. We compared the movie proposed by the SICS and the best 10 movies ranked by the CF algorithm computing the distribution in percentage. We run our implementation of the CF algorithm obtaining a Mean Absolute Error comparable to the ones presented in the literature [3, 8]. We run the experiments on the SICS changing the number of neighbours, the threshold and the functions used to compute the similarity among agents and actions. Table 1 presents the results of the experiments.

A low k implies a low number of scenes (items) valutated and consequently low performance (compare columns 1-6) because it is likely to miss some valuable items. With $T_{min} = 0$ the number of valutated scenes is on average 38.4 against 25.2 with $T_{min} = 0.8$, hence the speed of presentation is higher with a slight decrease in the performance (compare columns 4 and 7). The domain-dependent agent similarity function eq 4 is slightly better that the general one 1 (compare columns 4 and 8). Finally, a very simple action similarity (eq. 7) performs poorly (compare columns 4 and 9). We can conclude that with domain-dependent similarities our SICS is comparable with CF. More interestingly, comparing the last cells of column 4 and 8 it is possible to conclude that also the version without the domain-dependent agent similarity but with a general one performs satisfactorily.

5 An application: The eCulture Brokering System

In this section, we present the eCulture Brokering System [13], a multi-agent system for cultural information brokering where we have applied the SICS. The multi-agent system has been developed using JACK Intelligent Agents [4]. The system is an extension of the results of the MIRVAC project [18] developed by ITC-irst and University of Trento for the cultural domain of Trentino. The project has also introduced the innovative overhearing architecture [5], in which a suggester agent observes the messages that two or more parties exchange and tries to cooperate with them sending suggestions.

In a multi-agent system, a SICS can be either a general capability of the overall system or a specific capability of a single agent. In the former case, the SICS observes all the agents acting in the system and manipulates the environment. In the latter, the SICS is applied to what the agent is able to observe and change, namely the part of environment and the agents it interacts with. The SICS capability, both general and specific, affects the whole system. In the system we present here, we choose to adopt the second option where a SICS is a capability of a single agent. In order to gain effectiveness we embedded a SICS in a Directory Facilitator (DF), namely an agent that plays a central role in the interactions. In particular, we adopt the idea of DF from FIPA specifications [6] and we extends its capabilities with the SICS.

A DF is a mandatory agent of an agent platform that provides a yellow pages directory service to agents. Every agent that wishes to publicize its service to other agents, requests the registration to the DF providing a description of its services. An agent can ask the DF in order to request information about the services available and the agents registered for such services. By means of a SICS, the DF can produce the Implicit Culture relation between an agent and the agents that have previously requested information, and provide information that encounters the preference of the agent.

In particular, it focus on the agents interaction for which we use the SICS. The platform contains a set of personal agents `Pa_1`,...,`Pa_h`, a `DF` that provides information about a set of brokers `B_1`,...,`B_n` and a set of wrappers `W_1`,...,`W_m`. A personal agent is created and assigned to each user who accesses the system by means of a web browser. The brokers are specialized in providing information about a specific cultural area (for instance, history, archeology, art, etc...), and they can collect information from different wrappers. Each wrapper is built for a specific database of a museum of the Trentino region. Basically, the databases are of two types: Microsoft Access and Oracle. The complete architecture includes other agents, like for instance the agent resource broker, which provides information about the resources available outside the multi-agent system. The rule used to express the cultural theory is the following:

$$\begin{aligned}&\mathtt{request}(\mathtt{x},\mathtt{DF},\mathtt{s},\mathtt{t}_1) \wedge \mathtt{inform}(\mathtt{DF},\mathtt{x},\mathtt{y},\mathtt{t}_2) \wedge \mathtt{less}(\mathtt{t}_1,\mathtt{t}_2) \rightarrow \\ &\quad \mathtt{request}(\mathtt{x},\mathtt{y},\mathtt{s},\mathtt{t}_3) \wedge \mathtt{less}(\mathtt{t}_2,\mathtt{t}_3)\end{aligned} \tag{6}$$

that states that if `x` asks DF for the service `s`, and DF replays informing `x` that `y` can provide such a service, then `x` will request to `y` the service `s`.

The similarity function between two actions is the following:

$$sim(\beta,\alpha) = \begin{cases} 1 & if\ \beta = \alpha \\ 0 & otherwise \end{cases} \tag{7}$$

where $\beta = \alpha$ if the actions are of the same type (namely, `request` or `inform`) and have the same arguments.

The `DF` uses the SICS to suggest the broker to the personal agent. In particular, for each personal agent that sends a request, the `DF` finds all the agents that have previously performed similar actions (requests and consequent response messages), and then suggests the personal agent with the broker for which such agents would be satisfied. The experiments we have made have shown that the SICS can effectively improve the interaction among agents. In particular, it can help new agents (users), that do not know the domain, to interact with the multi-agent system.

6 Conclusions and future work

We have presented an implementation of a System for Implicit Culture Support that exploit an application of instance-based learning techniques. We have showed that, in a particular domain and with a simple a priori theory, the system is functionally equivalent to Collaborative Filtering. Moreover we have presented a real-world application. Our three-steps algorithm for proposing the scene can be considered a generalization of a Collaborative Filtering algorithm, where the similarity is performed on executed actions and not on ratings. This generalization is non–trivial for it puts the SICS in a wider framework than simple CF. In fact it is possible to vary the domain, the cultural contraint theory, and also to deal with artificial agents as we have shown with the eCulture system. A relevant portion of research on multi-agents learning [20] deals with reinforcement learning (e.g., [11]). From the point of view of reinforcement learning the works of Price and Boutillier [17] on *Implicit Imitation* are relevant to our work. The critical difference is that a SICS does not imitate but *induce* an agent to imitate or more generally to act consistently with another group of agents. In the broad area of web personalization a relevant work is the one by Paliouras et. al. [16] who clusters communities of on-line users. In our perspective the groups are given, consequently an integration of the methods would be interesting. Finally, our approach can be seen as a temptative of supporting organizational learning in the direction proposed in [21]. Future work will be devoted to experimentation on domains with a wider range of actions, more complex scenes and more complex, possibly induced, cultural constraint theories.

APPENDIX A: Formal Definition of Implicit Culture

We consider *agents* and *objects* as primitive concepts to which we refer with strings of type *agent_name* and *object_name*, respectively. We define the *set of agents* $\mathcal{P}$ as a set of *agent_name* strings, the *set of objects* $\mathcal{O}$ as a set of *object_name* strings and the *environment* $\mathcal{E}$ as a subset of the union of the set of agents and the set of objects, i.e., $\mathcal{E} \subseteq \mathcal{P} \cup \mathcal{O}$.

Let *action_name* be a type of strings, E be a subset of the environment ($E \subseteq \mathcal{E}$) and s an *action_name*.

Definition 3 (action). *An action α is the pair $\langle s, E\rangle$, where E is the argument of α ($E = arg(\alpha)$).*

Let $\mathcal{A}$ be a set of actions, $A \subseteq \mathcal{A}$ and $B \subseteq \mathcal{E}$.

Definition 4 (scene). *A scene σ is the pair $\langle B, A\rangle$ where, for any $\alpha \in A$, $arg(\alpha) \subseteq B$; α is said to be* possible in σ. *The* scene space $\mathcal{S}_{\mathcal{E},\mathcal{A}}$ *is the set of all scenes.*

Let T be a numerable and totally ordered set with the minimum t_0; $t \in T$ is said to be a *discrete time*. Let $a \in \mathcal{P}$, α an action and σ a scene.

Definition 5 (situation). *A situation at the discrete time t is the triple $\langle a, \sigma, t\rangle$. We say that a* faces *the scene σ at time t.*

Definition 6 (execution). *An* execution *at time t is a triple $\langle a, \alpha, t\rangle$. We say that a* performs *α at time t.*

Definition 7 (situated executed action). *An action α is a* situated executed action *if there exists a situation $\langle a, \sigma, t\rangle$, where a performs α at the time t and α is possible in σ. We say that a performs α in the scene σ at the time t.*

The action that an agent a performs at time t is a random variable $h_{a,t}$ that assumes values in $\mathcal{A}$.

Let $a \in \mathcal{P}$ and $\langle a, \sigma, t\rangle$ be a situation.

Definition 8 (expected action). *The* expected action *of the agent a is the expected value of the variable $h_{a,t}$, that is $E(h_{a,t})$.*

Definition 9 (expected situated action). *The* expected situated action *of the agent a is the expected value of the variable $h_{a,t}$ conditioned by the situation $\langle a, \sigma, t\rangle$, that is $E(h_{a,t}|\langle a, \sigma, t\rangle)$.*

Definition 10 (party). *A set of agents $G \subseteq \mathcal{P}$ is said to be a* party.

Let $\mathcal{L}$ be a language used to describe the environment (agents and objects), actions, scenes, situations, situated executed actions and expected situated actions, and G be a party.

Definition 11 (cultural constraint theory). *The* Cultural Constraint Theory for G *is a theory expressed in the language $\mathcal{L}$ that predicates on the expected situated actions of the members of G.*

Definition 12 (group). *A party G is a* group *if exists a cultural constraint theory $\mathbf{\Sigma}$ for G.*

Definition 13 (cultural action). *Given a group G, an action α is a* Cultural Action w.r.t. G *if there exists an agent $b \in G$ and a situation $\langle b, \sigma, t\rangle$ such that*

$$\{E(h_{b,t}|\langle b, \sigma, t\rangle) = \alpha\}, \mathbf{\Sigma} \not\vdash \bot$$

where $\mathbf{\Sigma}$ is a cultural constraint theory for G.

Definition 14 (implicit culture). Implicit Culture *is a relation* $\rtimes\!\!\!\ltimes$ *between two parties* G *and* G' *such that* G *and* G' *are in relation* $(G \rtimes\!\!\!\ltimes G')$ *iff* G *is a group and the expected situated actions of* G' *are cultural actions w.r.t* G.

Definition 15 (implicit culture phenomenon). Implicit Culture Phenomenon *is a pair of parties* G' *and* G *related by the Implicit Culture.*

We justify the "implicit" term of implicit culture because its definition makes no reference to internal states of the agents. In particular, there is no reference to believes, desires or intentions and in general on epistemic states or to any knowledge about the cultural constraint theory itself or even the two composition of the two groups. In the general case, the agents do not anything explicitly in order to produce the phenomenon.

References

1. D. Billsus and M. Pazzani. Learning collaborative information filters. In *Proceedings of the International Conference on Machine Learning*, Madison, Wisciautoh, 1998. Morgan Kaufmann Publishers.
2. Enrico Blanzieri and Paolo Giorgini. From collaborative filtering to implicit culture. In *Proceedings of the Workshop on Agents and Reccomender Systems*, Barcellona (http://www.science.unitn.it/ pgiorgio/ic).
3. J. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, Madison, WI, July 1998. Morgan Kaufmann.
4. P. Busetta, R. Rönnquist, A. Hodgson, and A. Lucas. Jack intelligent agents - components for intelligent agents in java. AOS TR9901, January 1999. http://www.jackagents.com/pdf/tr9901.pdf.
5. P. Busetta, L. Serafini, D. Singh, and F. Zini. Extending multi-agent cooperation by overhearing. ITC-IRST Technical Report 0101-01, 2001. http://sra.itc.it.
6. FIPA. *Foundation for Intelligent Physical Agents.* http://www.fipa.org.
7. Enrico Blanzieri Paolo Giorgini and Fausto Giunchiglia. mplicit culture and multi-agent systemss. In *Proceedings of the International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet, (SSGRR 2000)*, L'Aquila - Italy (http://www.science.unitn.it/ pgiorgio/ic), 2000.
8. A. Gokhale. Improvements to collaborative filters algorithms. Master's thesis, Worchester Ploytechnic Institute, May 1999.
9. D. Goldberg, D. Nichols, B.M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.
10. J. Herlocker, J. Konstan J., A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 1999 Conference on Research and Development in Information Retrieval*, 1999.
11. J. Hu and M. Wellman. Multiagent reinforcement learning theoretical framework and an algorithm. In *Proceedings of the ICML*, Madison, Wisciautoh, 1998. Morgan Kaufmann Publishers.
12. J.A. Konstan, B.N. Miller, D. Maltz, J.L. Herlocker, L.R. Gordon, and J. Riedl. Grouplens: Applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87, 1997.

13. E. Blanzieri P. Giorgini P. Massa and S. Recla. Implicit culture for multi-agent interaction support. In *Proc. of the Int. Conf. on Cooperative Information Systems COOPIS*, 2001.
14. P. McJones. Eachmovie collaborative filtering data set, dec systems research center, 1997. http://research.compaq.com/SRC/eachmovie/.
15. I. Nonaka and H. Takeuchi. *The knowledge Creating Company.* Oxford University Press, New York, 1995.
16. G. Paliouras, C. Papatheodorou, V. Karkaletsis, and C. Spyropoulos. Clustering the users of large web sites into communities. In *Proceedings of the ICML*, Stanford, 2000. Morgan Kaufmann Publishers.
17. B. Price and C. Boutilier. Implicit imitation in multiagent reinforcement learning. In *Proceedings of the ICML*, Bled, 1999. Morgan Kaufmann Publishers.
18. MIRVAC Project. *ITC-IRST project.* http://sra.itc.it/project.epl?name=Mirvac.
19. Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM CSCW'94 Conference on Computer-Supported Cooperative Work*, 1994.
20. P. Stone and M. Veloso. Multiagent systems: A survey from amachine learning perspective. *Autonomous Robots*, 8(3):345–383.
21. K. Takadama, T. Terano, and K. Shimohara. Can multiagents learn in organization? – analyzing organizational learning-oriented classifier system. In *IJCAI'99 Workshop on Agents Learning about, from and other Agents*, 1999.

Is it all About Connections? Factors Affecting the Performance of a Link-Based Recommender System

Miles Efron and Gary Geisler
{efrom, geisg}@ils.unc.edu
School of Information and Library Science
University of North Carolina, Chapel Hill *

August 23, 2001

Abstract

This study reports on a recent evaluation of the similarity model used by Recommendation Explorer, an automatic recommender system. In particular, we consider the role of several system-internal factors in determining the quality of recommendation. More generally, we discuss factors in the recommendation task itself that complicate the construction and evaluation of recommender systems, and reflect on the implications of our findings for research in this area.

1 Introduction

Evaluating information retrieval systems is a notoriously steep challenge. The subjectivity of such crucial variables as the relevance and quality of information complicates the matter of evaluation. Automatic recommender systems are similar to IR systems in many ways [10, 11]. Among these similarities, the problem of evaluation is particularly vexing [14]. In this paper we focus on the evaluation problem by analyzing the factors that affect the performance of Recommendation Explorer, an automatic recommender system.

Borrowing from bibliometrics and hypertext analysis, Recommendation Explorer (RecEx) operates on a square matrix that describes the paths of inter-recommendation between database items. Due to the conceptual similarity between recommendation, citation, and hypertextual linking, we refer to RecEx as a "link-based" system. *Link* here suggests the type of relationship found in bibliometric or hypertextual structures. Thus our use of the term differs somewhat from the graph-theoretic sense used in other research [1, 16].

RecEx applies the technique of singular value decomposition (SVD) to the square item-item recommendation matrix. SVD allows evocative high-order semantic patterns to inform the system's similarity model. In previous work Sarwar *et al.* [13, 12] used SVD to obtain robust recommendations from sparse correlational data. Our work differs from theirs insofar as we use SVD to analyze inter-item relationships, not relations between users.

In a previous study [7] this method of recommendation yielded promising results. The present study thus focuses on problems of interpreting the results of conventional evaluation methods. In particular, we analyze system performance in close detail to discover not only how well our system works, but when it works well and when it fails. We detail several aspects of the recommendation task that bear on the quality of our system's performance, reflecting on their implications for system design and evaluation.

*This research was funded in part by the School of Information and Library Science at the University of North Carolina, and by Ibiblio.org

Figure 1: User-defined interest profiles enable quick, personalized recommendations

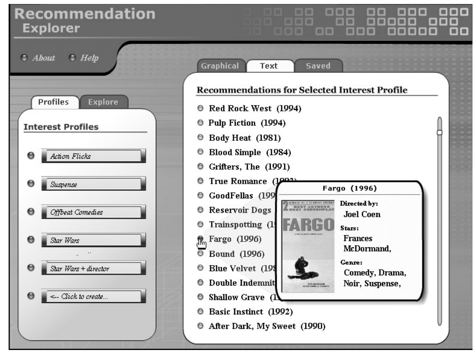


Figure 2: Interface for viewing and updating recommendations

2 Recommendation Explorer

Recommendation Explorer (RecEx) is an experimental recommender system under development at the School of Information and Library Science at The University of North Carolina, Chapel Hill. In its current implementation RecEx uses a database of 12,726 popular film titles. Each film in the database is represented by a metadata record that contains a plot summary, production information, and a list of other films that human editors have recommended for those who like the film.

Like all recommender systems, RecEx faces a steep challenge: to accommodate the multidimensional and dynamic nature of recommendation. The demands that each user puts on a recommender system are highly specific and subject to change. Accounting for such complexity in user motivations, expectations, and context is difficult, but critical for a system that provides personalized recommendations. RecEx addresses this difficulty by tightly coupling the system's interface with algorithms that derive a powerful model of inter-item similarity.

Suppose that user A likes *The Usual Suspects* because of its stars, while user B likes it because of its style. The interface to RecEx allows each user to identify these axes of his or her information need. Using stored, named profiles (Figure 1), the user specifies items that he likes and the item attributes that are important to him. After recommendations are generated from the profile parameters, the user can view and explore the results, previewing and saving items and developing an understanding of the results (Figure 2). This understanding might lead the user to adjust the expression of his information need.

This type of interaction depends upon a robust similarity model. That is, if user A liked *The Usual Suspects* because of its actors, the system needs a way to find actors *similar* to those in *The Usual Suspects*. A model of film-film similarity is important because it allows us to define similar actors, directors, etc: e.g. similar actors act in similar films.

The remainder of this paper describes the construction and evaluation of an experimental module of RecEx. This module implements an inter-item similarity model based on the singular value decomposition of the film-film recommendation matrix. The similarity module derives a mapping of the system's recommendation space, collocating "similar films." We devote particular attention to the problems of evaluating such a module.

3 The RecEx Similarity Model

The database behind RecEx derives from the recommendation service of reel.com, a database of movie information on the Internet. For each film in the database, we record two sets of information: 0 or more close recommendations and 0 or more creative recommendations. A close recommendation from film A to film B implies an obvious link between the two, such as a common director or subject matter. A creative recommendation describes a more tenuous relationship. All films in the database have at least one of the following: close recommendation, creative recommendation, incoming recommendation (i.e. recommended by another film). On average, each film contains 4.035 recommendations, 2.294 close and 1.741 creative.

These recommendations were compiled by human editors of the reel.com database. The proposed method attempts to exploit the expertise of these editors to the greatest extent possible. However, other types of data could inform a system such as we describe. In an e-commerce setting, an item-item matrix $\mathbf{A}$ could be constructed wherein each cell a_{ij} contains a count of the number of times item i was purchased in the same order as item j. A digital library might create a matrix from the citation or hyperlink patterns among documents.

To define item-item similarity we begin with the square film-film recommendation matrix $\mathbf{A}$ where each cell a_{ij} records the type of recommendation that the *ith* film makes regarding the *jth* film. If film j is a close recommendation for i, $a_{ij} = 2$. If film m is a creative recommendation for i, $a_{im} = 1$. We also set cells on the main diagonal equal to 5. Thus $a_{ii} = 5$ (these values were chosen because they led to good performance).

In its original implementation RecEx worked directly on this matrix. For a given seed film i, the system returned N recommendations by recursively following the links from i supplied by the reel.com editors. We refer to this method as the use of the "raw" link structure. To generate robust similarity judgements, the new method transforms the film-film matrix $\mathbf{A}$ through application of the singular value decomposition (SVD).

3.1 Singular Value Decomposition: Motivation

Used widely in information retrieval (where it goes by the name latent semantic indexing, or LSI) [3, 2] SVD is a least-squares dimensionality reduction technique. A type of factor analysis, SVD is closely related to principal components analysis and multidimensional scaling. The goal of LSI is to represent items along axes that manifest the "latent semantic structure" of a matrix. To accomplish this LSI uses SVD to project a matrix $\mathbf{A}$ of rank r onto a space of k dimensions, where $k \ll r$. The resulting k-dimensional matrix $\mathbf{A}_k$ is the closest rank-k approximation of $\mathbf{A}$, in the least squares sense. Its proponents argue that this projection into k-space reduces noise in the matrix $\mathbf{A}$.

In information retrieval, use of LSI is motivated by the suspicion that lexical features provide noisy evidence about document relationships. While lexical ambiguity is not an issue in our film-film recommendation matrix, SVD is still valuable for RecEx. This is due to SVD's analysis of high-order relationships between matrix elements. If film i recommends film j, and j recommends m, our system will recognize a transitive affinity between i and m.

3.2 Singular Value Decomposition: Mathematics

To compute the singular value decomposition[1], we begin with the film-film matrix $\mathbf{A}$, described above (in our case $\mathbf{A}$ is square, but it need not be). During the SVD, our $n \times n$ matrix $\mathbf{A}$ of rank r is factored into the product of three special matrices (Formula 1).

$$\mathbf{A} = \mathbf{T}\mathbf{\Sigma}\mathbf{D}' \tag{1}$$

Matrices $\mathbf{T}$ and $\mathbf{D}$ are orthonormal: $\mathbf{T}'\mathbf{T} = \mathbf{D}'\mathbf{D} = \mathbf{I}_n$ and the columns of $\mathbf{T}$ and $\mathbf{D}$ are of unit length. $\mathbf{T}$ and $\mathbf{D}$ comprise the left and right singular vectors of $\mathbf{A}$, respectively. The $r \times r$ diagonal Matrix $\mathbf{\Sigma}$ contains the singular values of $\mathbf{A}$ in descending order on the main diagonal. The

[1] A full description of SVD is beyond the scope of this study. For a more detailed treatment see [9]

singular values are the positive square roots of the eigenvalues of $\mathbf{A}'\mathbf{A}$ and $\mathbf{A}\mathbf{A}'$. Thus the ith singular value indicates how much of the input matrix's variance is described by the ith axis of factor space.

Matrix $\mathbf{T}$ represents the rows of the original matrix $\mathbf{A}$. Thus the ith column of $\mathbf{T}$ describes the ith film as a vector in factor space. $\mathbf{D}$ represents the columns. In the case of the Recommendation Explorer, $\mathbf{T}$ and $\mathbf{D}$ are equivalent.

The dimensionality reduction in LSA comes about by truncating the matrix $\mathbf{\Sigma}$ and then recombining it with the matrices $\mathbf{T}$ and $\mathbf{D}$. Because SVD by definition will find r factors for matrix $\mathbf{A}$ where $rank(\mathbf{A}) = r$, as we approach the rth factor, the amount of variance described by each axis will be very small. Because the last singular values are small, we suspect that they represent noise, that they describe random variance. By choosing a dimensionality k, setting all singular values i for $i > k$ equal to 0, and amending $\mathbf{T}$ and $\mathbf{D}$ accordingly, by matrix multiplication we project $\mathbf{A}$ onto the best $k-$dimensional space, in the least-squares sense.

3.3 Implementation

To compute the SVD of our film-film matrix, we used SVDPACKC [8], a suite of programs for solving eigensystems of sparse matrices. After computation, we project the left singular vectors into $k-$space. Similarity judgements are performed on the matrix $\hat{\mathbf{T}} = \mathbf{T_k}\mathbf{\Sigma_k}$, where $\mathbf{T_k}$ contains the first k rows of $\mathbf{T}$ and $\mathbf{\Sigma_k}$ is the $k\times$k matrix defined by the first k rows and columns of $\mathbf{\Sigma}$.

The matter of choosing an optimal k value is an open question in the LSA research [4, 15]. Common practice in IR applications indicates a dimensionality between 50 and 300. After some trial and error, we selected $k = 50$.

$$cos(\overrightarrow{u}, \overrightarrow{v}) = \frac{\overrightarrow{u} \cdot \overrightarrow{v}}{||\overrightarrow{u}||\,||\overrightarrow{v}||} \tag{2}$$

Similarity between two films v and u is thus defined as the cosine (Formula 2) of each film's vector in the k space defined by $\hat{\mathbf{T}}$.

4 Experimental Evaluation

To gauge the effectiveness of SVD for our application we conducted an experiment to compare the performance of the SVD module against performance based on the raw link structure defined in matrix $\mathbf{A}$.

4.1 Methodology

Evaluation was conducted using 10 films, listed in Table 1. These "seed" films were chosen to represent a variety of film genres and audience types.

Evaluation occurred in two phases. In the first phase, volunteer reviewers defined a generous list of recommendations for each seed film. Six reviewers were chosen from a sample of convenience, based on their self-identified interest in popular films. Each reviewer chose the 5 seed films on which he felt most competent to make recommendations. For each chosen seed, each reviewer created a list of approximately 30 candidate recommendations. These lists were then combined to create a "recommendation space" for each seed. The largest pooled space was for *Austin Powers*, with 75 members. The smallest was *The Terminator*'s 37. The mean recommendation space was 51.9 titles; the median was 51.

To help them compile their 30-film list for each seed, reviewers used three online resources: The Internet Movie Database (http://www.imdb.com), The Movie Critic (http://www.moviecritic.com), and The Sepia Video Guide (http://vguide.sepia.com). IMDB provides recommendations that derive from several sources: user suggestions, IMDB editor picks, and an undisclosed automatic system. The Movie Critic uses the LikeMinds collaborative filtering system. Sepia is a simple online reference work that contains information about films, but does not make explicit recommendations. In addition, reviewers were permitted to add any titles to the list not found in the online systems.

In the second evaluation phase, 131 new reviewers picked approximately 15 recommendations for each of several seed films. On average these 131 reviewers made recommendations for 3.9 seeds. *Fargo* received the most reviews, 68. *Spanish Prisoner* received the fewest, with 19. The average number of reviews per

Title	Raw	SVD
alien	0.287	0.435
austin powers	0.240	0.431
english patient	0.048	0.061
fargo	0.247	0.373
full monty	0.192	0.048
room view	0.256	0.456
sleepless	0.122	0.185
spanish prisoner	0.233	0.265
star wars	0.471	0.364
terminator	0.484	0.403

Table 1: Average Precision of recommendations based on raw link structure and links analyzed by SVD

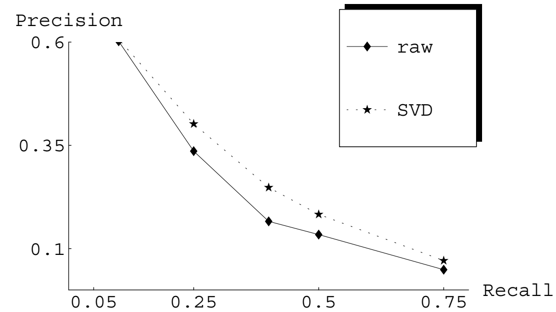


Figure 3: Average precision/recall for the raw data and data transformed by SVD

seed was 51.4, with a median of 54.

Using an online form, each reviewer consulted the pooled recommendation space for each of his selected seeds, marking all those films that he had seen and the 10-15 best recommendations for fans of the seed.

Finally, these reviews were pooled into a "key" for each seed—a list of films that constitute good recommendations for a given seed. The key contains all candidate films selected by any reviewer. Using our key as a point of reference, we evaluate recommendation performance in terms of precision and recall. Precision is defined as the ratio of the number of returned key-members to total films retrieved (percent of returned items that are relevant). Recall is the ratio of the number of key-members retrieved to the total number total key members (percent of all relevant items returned).

4.2 Results

4.2.1 Quality of SVD recommendations

Table 1 compares the two methods, raw and SVD, using average precision at five levels of recall (.1, .25, .4, .50, .75). The SVD method provides better average precision for seven out of the ten seed films. Figure 3 plots precision against recall for each method, with each recall/precision point averaged across all ten seeds. SVD appears to offer the most improvement over the raw link structure at middling levels of recall. This result is somewhat surprising, as we anticipated that SVD would yield greater performance improvements for high-recall searches. A partial explanation for the tapering off of SVD's benefit at high recall may lie in the nature of the task imposed by this experiment. Since candidates are only counted as "relevant" if they appear in the user-defined key, we may be penalizing recommendations that would be perfectly viable to a user, but that were simply not included in the seed key. This amounts to a problem of defining the set of all "relevant" candidates for a given seed, a problem to which we return in the following section.

4.2.2 Dimensionality of the Similarity Model

To gauge how the choice of k, the dimensionality of the reduced space, bears on recommendation quality, we generated spaces of varying dimensionality and measured precision/recall using each space.

Figure 4 charts a dramatic improvement in precision performance as we increase k from 45 to 50. As k increases from 50 to 100, performance degrades slightly. Spaces of dimensionality much lower than 45 appear insufficiently informative for the recommendation task, while the 100-dimensional space is slightly less effective than the 50-dimensional space.

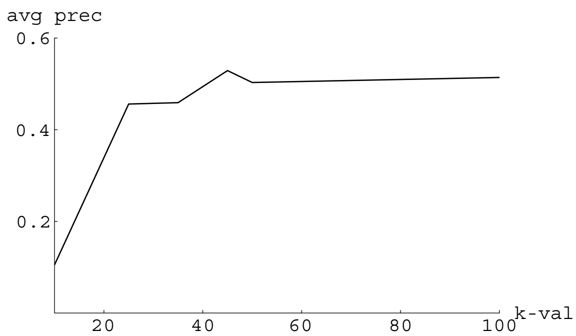


Figure 4: Average precision in terms of the dimensionality of the SVD space.

4.3 Factors Affecting Recommendation Quality

Having analyzed our data, it is clear that the similarity model matches the tastes of our volunteer reviewers better for some seed films than others. For example, average precision for *The English Patient* was extremely low, while the similarity model delivered fairly good precision for *A Room with a View*. This section pursues an analysis of which factors bear on this discrepancy in performance.

A series of linear models were constructed to measure the effects of several independent variables on system performance. In each case, performance for a given seed is modeled as a linear combination of factors. The goal of this modeling is to gain an appreciation for why certain seeds led to decreased or increased performance in our experiment. In particular we were eager to discover how much variability in performance derives from defects in our similarity model, and how much is due to other causes.

Because the SVD-based similarity model depends on the link structure defined by patterns of inter-item recommendation we suspected that performance would be affected by the degree to which a given seed film is linked to other films. To assess this effect we defined for each film in the database two data–its in-degree and out-degree. The in-degree for film A counts the number of films that recommend A. Likewise, A's out-degree describes the number of films that A recommends. Not surprisingly most popular films have a higher in-degree than out-degree. For example 35 films recommend *Alien*, while *Alien* recommends 9. For the sake of comparison, we included as a seed film *The Spanish Prisoner*, with in-degree of only 1 and out-degree of 9.

Formula 3 describes the linear model that was constructed to gauge the relation between performance and a seed film's degree of connection to the database recommendation structure:

$$y_i = \beta_0 + \beta_1(in_i) + \beta_2(out_i) + \beta_3(out/in_i) + \epsilon_i \quad (3)$$

where y_i=average precision for seed i. To our surprise, however, this model fit the data very poorly, yielding R^2=0.119. The model describes less than 12% of the variance among seed film performance. The p-value for the null hypothesis H_0 : *no linear relation adheres among the variables* was 0.845. We supplemented this analysis with a second model, identical to the prior model with the exception that in this case we replaced y_i with precision at 25% recall, rather than average recall. The second model also suffered from a poor fit, with R^2=0.182 and p-value=0.73.

These findings indicate that in-degree and out-degree bear very little on the performance for the observed data. Instead, we suspected that variations in precision might derive from qualities native to the seed films themselves. Perhaps certain films do not lend themselves to comparison by our similarity model. For such films, the idea of 100% recall might not be meaningful. In other words, perhaps certain seeds are simply harder to provide general recommendations for than others. To help measure the "difficulty" of recommending for a given seed, we consider three variables. *Reviewers* counts the number of participants who reviewed a given seed A. Seed film A's *total* records the total number of reviews generated for A during the experiment. Thus *total* divided by *reviewers* yields the average number of films reviewed by a single participant for seed A. Finally, *unique* measures the number of distinct candidate films recommended for A.

Using these variables we constructed another model, shown in formula 4:

$$y_i = \beta_0 + \beta_1(reviewers_i) + \beta_2(total_i) + \beta_3(unique_i) + \epsilon_i \quad (4)$$

where y_i=average precision for seed i. This model yielded a much better fit than the previous models, with R^2=0.64. The p-value for H_0 was 0.087. Using y_i=precision at 25% yielded R^2=0.742, with p-value=0.034. Encouraged by the fit of these models, we plotted the residuals against the predicted values for each. Data in both plots appeared to be distributed randomly, suggesting that little or no systematic variance is omitted by the new models.

4.4 Assessing the variability of Observed Statistics

An obvious limitation of the modeling described here lies in the small number of seeds in the study. Due to our low N, a high degree of variability must be expected in trying to generalize our results. To gain a sense of this variability we conducted a round of bootstrap resampling. Bootstrapping (described in [5, 6]) is a computer-intensive method for assessing the variability of a statistic S. The process involves repeated resampling from the original data, and concomitant re-computation of the statistic of interest. For a data set D of size N, we create M bootstrap samples D^* by sampling N observations from D, with replacement. For each bootstrap sample D^*, we compute our statistic S^*. By repeating this process M times, where M is large, we can estimate the stability of S by noting the variability of S^*.

For this application we set M=1000, thus creating 1000 bootstrap samples from our original data. For each of these 1000 bootstrap samples, we computed R^{2*}. We then calculated the standard deviation of R^{2*}. For the model where y_i=average precision, mean R^{2*}=0.793, and $s(R^{2*})$=0.161. With y_i=precision at 25% recall, mean R^{2*}=0.633, and $s(R^{2*})$=0.305. Although our observed R^2 is quite variable we can still glean useful information from our models. If we take $s(R^{2*})$=0.161 to be an estimate of the standard deviation of the true R^2, it is highly unlikely that the true value of R^2 is less than 0.632 (i.e. two standard deviations away from the observed $\hat{R}^2$). Thus it seems that variables *reviewers*, *total*, and *unique* do account for a sizable portion of the performance variation among seed films.

A potential weakness of the link-based similarity model used by RecEx is that weak patterns of co-recommendation in the data might limit performance. The relationship between performance and seed connection measures reported above, however, indicate that this is not the case. Our method appears to discern film-film similarity regardless of the in- and out-degree of seed films.

A strong correlation was observed between system performance and measures of reviewer agreement. This finding suggests that in some cases a similarity model based on static, high-level relationships between films cannot suitably address the complexity of individual users' tastes. In the case of *The English Patient*, for instance, many of the reviewers' top candidates were films based on romantic novels such as *Sense and Sensibility*, *Howard's End*, and *Emma*. On the other hand, other *English Patient* reviewers selected films with Ralph Fiennes (*The English Patient's* co-star) as an actor. Finally, many recommendations for *The English Patient* were period pieces. *The English Patient* describes events that occurred during World War II and before. Thus many reviewers recommended films made during the 1940s, or films that depict that period. On the other hand, the similarity model was unable to pursue these alternative types of interest in the film, recommending an array of "epic" stories, linked more by the scope of their narrative than any single, concrete factor. SVD recommendations for *The English Patient* are general in the face of a recommendation space that is highly specialized and disjointed.

Our results indicate that even a strong model of item-item similarity will have difficulty satisfying information needs as diverse as those evidenced by *The English Patient's* reviewers. To improve recommendation quality we propose supplementing static similarity judgements with user-specific information that "fine-tunes" the input to the recommendation system. Such tuning might include information concerning aspects of seed films that are important to the user at query time.

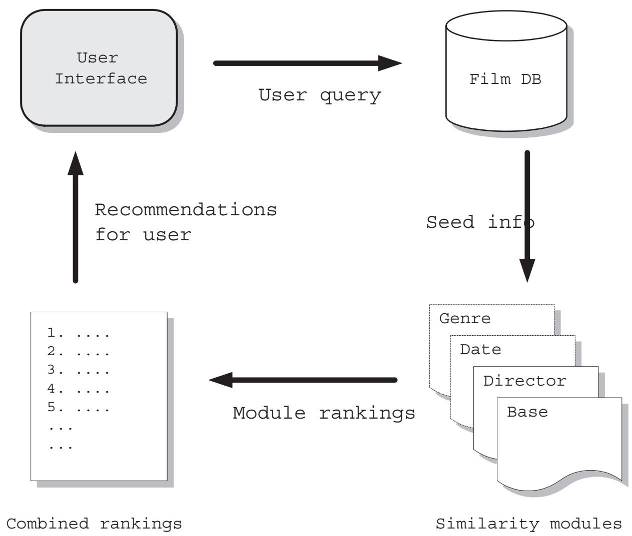


Figure 5: Modular Architecture of RecEx System

As an example, RecEx enables users to specify not only which movies they like, but also why they like them. A user might request movies similar to *The English Patient*, for instance, but also indicate that director and date are important to him. These supplementary similarity judgments are made by separate modules of the system (Figure 5). The director module, for instance, defines similarity between directors in terms of film-film similarity: similar directors make similar movies. This modular approach capitalizes on the film-film similarity model in order to deliver personalized recommendations with minimal user effort.

5 Conclusion

Recommendation Explorer's model of item-item similarity uses the Singular Value Decomposition to discover high-order relationships between database items. Based on patterns of co-recommendation among films, a system based on this similarity model has yielded promising results for automatic recommendation. A potential defect of such a link-based model (poor recommendation for weakly connected items) was not observed in the study reported here. However, this study suggests that even a robust similarity model can be supplemented by data specific to a given user at a given time. A detailed analysis of system performance suggests that human motivations in recommendation are diverse and highly specific. We anticipate improved performance after supplementing the static film matching module described here with modules that permit more nuanced articulations of information needs. Involving the more actively in the recommendation process, we propose, will improve the utility of our model of inter-item similarity. However, the diversity of reviewer opinion found in this study calls into question methods of evaluating such complex systems. Thus the matter of recommender evaluation itself remains a matter to be evaluated.

References

[1] Charu Aggarwal, Joel Wolf, Kun-Lung Wu, and Philip Yu. Horting hatches an egg: A new graph-theoretic approach to collaborative filtering. In Surajit Chaudhuri and David Madigan, editors, *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 201–212, N.Y., August 15–18 1999. ACM Press.

[2] M. W. Berry, Susan T. Dumais, and G. W. OBrien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4):573–595, 1995.

[3] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *J. of the American Society for Information Science*, 41(6):391–407, 1990.

[4] C. H. Q. Ding. A similarity-based probability model for latent semantic indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1999.

[5] B. Efron. Bootstrap methods: another look at the jackknife. *Annals of Statistics*, 7:1–26, 1979.

[6] B. Efron. *An Introduction to the Bootstrap.* Chapman and Hall, 1993.

[7] M. Efron and G. Geisler. Using dimensionality reduction to improve similarity judgements for recommendation. In *Proceedings of the Second DELOS Network of Excellence Workshop on 'Personalisation and Recommender Systems in Digital Libraries'*, number No. 01/W03 in ERCIM Workshop Proceedings, 2001. http://www.ercim.org/publication/ws-proceedings/DelNoe02/index.html.

[8] M. W. Berry *et al.* Svdpackc (version 1.0) user's guide, university of tennesse tech. report cs-93-194. Technical report, University of Tennessee, 1993 (Revised October 1996). Available at http://www.netlib.org/svdpack/index.html.

[9] G. H. Golub and C. F. van Loan. *Matrix Computations.* Baltimore, The Johns Hopkins University Press, 1989.

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

[11] D. Riecken. Introduction: personalized views of personalization. *Communications of the ACM*, 43(8):26–28, 2000.

[12] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of recommender algorithms for e-commerce. In *ACM E-Commerce 2000 Conference*, October 2000.

[13] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender systems–a case study. In *ACM WebKDD Workshop*, 2000.

[14] A. F. Smeaton and J. Callan. Joint delos-nsf workshop on personalisation and recommender systems in digital libraries. In *ACM SIGIR Forum*. Association for Computing Machinery, 2001. Forthcoming.

[15] R. E. Story. An explanation of the effectiveness of latent semantic indexing by means of a bayesian regression model. *Information Processing and Management*, 32(3):329–344, 1996.

[16] L. Terveen and B. Amento. Constructing, organizing, and visualizing collections of topically related web resources. *ACM Transactions on Human-Computer Interaction*, 6(1):67–94, 1999.

Evaluation of integrated content-based collaborative filtering

Kaname Funakoshi and Takeshi Ohguro*
NTT Communication Science Laboratories, NTT Corp.
2-4 Hikaridai, Seika, Soraku, Kyoto 619-0237 Japan
kf@cslab.kecl.ntt.co.jp

Abstract

In this paper, we present a system evaluation of a content-based collaborative information filtering method called Nakif, which uses user evaluation data and item content data together. Although Nakif is originally an incremental collaborative filtering approach that modifies user profiles while obtaining evaluations, we additionally introduce a non-incremental profile construction strategy. Both the incremental and non-incremental methods are measured by using the MovieLens dataset, which includes evaluation data by real users to real movies. The results show that the content-based collaborative filtering approach can provide information items efficiently.

1 Introduction

Collaborative filtering is an information search technique that uses the evaluations of other users to provide (or recommend) information items [3]. The aim of collaborative filtering is to search for high-quality information items or items that match a user's ambiguous tastes, which cannot be handled by classical content-based information filtering.

There are various methods of collaborative filtering each with its own objectives. For items of absolutely measurable qualities, only evaluations given by the authorities of the domain concerned are useful. In other cases, reference can be made to the large numbers of evaluations given by anonymous people.

In domains where the qualities of items cannot be decided absolutely, e.g., music, art, novels, movies, etc., items that match the user's tastes should be recommended. To select items based on a user's tastes, it is better to refer to the relations of evaluation tendencies of every two users. In these cases, methods that use the correlations between users' evaluations (correlation-based methods) would be useful for item selection.

*Currently with NTT-ME Corp.

However, simple correlation methods have several disadvantages in specific domains, e.g., they have to handle the content information of each item.

We earlier suggested a content-based collaborative filtering method that uses the efficiency to other users for each keyword which represents the content of an information item [2]. In our previous study, the method was tested by simulation on a set of journal articles and a set of virtual users. Unfortunately, there were at least two problems in the simulation. First, it was hard to assume those virtual users with ambiguous tastes. Second, the process of selecting journal papers was not suited for collaborative filtering considering the tastes of users. We therefore felt it necessary to test our method under a more realistic situation.

In this paper, we present a development of our method and system evaluations on the performance of content-based collaborative filtering on the MovieLens dataset, a collection of evaluation data of real users to real movies presented by the GroupLens project [4].

The handling of movies is an appropriate domain for collaborative filtering because the appropriate selection of items depends on the tastes of users and the quality levels of items, ambiguous standards hardly able to be handled by content-based recommendation. This implies that content-based methods can be implemented in the field of movie selection.

In the next section, we explain collaborative filtering briefly, especially content-based collaborative filtering. Then, in the following sections, we present evaluations of filtering systems and show the obtained results.

2 Collaborative filtering

In general, collaborative filtering uses evaluations to information items given by others. In collaborative filtering, all evaluations should not be handled equally; how effective an evaluation is depends on who the evaluator was, when the evaluation was done, etc. The effectiveness of each evaluation is decided in a uniform manner by the system or decided for each user according to the domain of the information item or objectives of the system.

This work presents methods where each user has a profile that is used to weigh each evaluation. This is because we handle the user's ambiguous tastes rather than the quality levels of items which can be measured by the numbers of evaluations or evaluations by authorities.

In this section, we explain three types of methods, i.e., a correlation-based method, Nakif, and non-incremental Nakif, where each user has a weight (efficiency) for each evaluation and the evaluation histories of every two users are used.

2.1 Correlation-based collaborative filtering

In a correlation-based method, the effectiveness of each evaluation is calculated by using the weight array of a user to another. The weight array is calculated by using the correlations of the user's evaluation tendencies.

Item h's relevance (prediction) to a user i is calculated by a prediction function $pred(i,h)$ as follows.

$$pred(i,h) = \bar{ev}_i + \frac{\sum_{j=1}^{N_{user}} (ev_{jh} - \bar{ev}_j)\ weight_{ij}}{\sum_{j=1}^{N_{user}} weight_{ij}} \tag{1}$$

where $weight_{ij}$ is an element of the user's weight array constructed by the similarities between the evaluations of user i and user j, ev_{ih} is the evaluation value given by user i to item h, $\bar{ev}_i$ is the mean of the evaluation values given by user i, and n is the number of neighbors having a high similarity with user i. The Pearson correlation coefficient is an effective method for constructing the matrix of $weight$ [5].

$$weight_{ij} = \frac{\sum_{h=1}^{N_{item}} (ev_{ih} - \bar{ev}_i)\ (ev_{jh} - \bar{ev}_j)}{\sigma_i\ \sigma_j} \tag{2}$$

However, a correlation-based method is not always appropriate for collaborative filtering. For example, content information is not mentioned in simple correlation-based methods. A good evaluator in one category may not be a good evaluator in another category; a friend who has very similar preferences in classical music is not guaranteed to have similar preferences in folk music. Content information should be implemented in domains where various kinds of items are exchanged. Accordingly, although the Pearson correlation coefficient is not a content-based collaborative filtering method, the performance of the approach shows that it is worth using in evaluations.

2.2 Nakif

We earlier suggested a content-based collaborative filtering method named Nakif to select information items based on the confidence to evaluators for each keyword [2].

In Nakif, the general model of collaborative filtering is described as follows. Here, each item has a profile (*document profile*) to describe its feature. Each user also has a profile (*user profile*) to describe his/her features for information selection. The facility for calculating the relevance of an information item to the user is called *matching* and it uses a function (called the *matching function*) to calculate the relevance between the user profiles and document profiles. Information items are presented to the user according to these relevance values. After receiving the information items, the user views the items, evaluates them, and then puts evaluation values into the system. When the system receives these evaluation values from the user, it executes the modification process to the user's profile, document profile, and in some cases, matching functions. The modification facility is called *feedback*.

Next, we briefly describe the procedure of Nakif. In Nakif, the document profile of an information item h is represented as a pair of vectors $\langle \boldsymbol{kw}_h, \boldsymbol{ev}_h \rangle$, where $\boldsymbol{kw}_h = (kw_{h1}, \ldots, kw_{hN_{term}})$ is a keyword vector describing the content of the corresponding item and $\boldsymbol{ev}_h = (ev_{h1}, \ldots, ev_{hN_{user}})$ is an evaluation vector describing the evaluation value from the user. N_{term} is the number of keywords and N_{user} is the number of users of the system. In the original Nakif, each evaluation has three real values: *good* for a positive evaluation, *bad* for a negative evaluation, and ϵ for neutral (not evaluated yet) where $good > bad$. A five-point scale is used in an evaluation data set as we will see in Section 3. Therefore, the values of an element of an evaluation vector should be $ev_{hi} = \{2, 1, 0, -1, -2, \epsilon\}$ for given values $\{5, 4, 3, 2, 1, \epsilon\}$ by subtracting the midpoint value.

Each user's profile is described as an array of confidence values to the evaluations of other users on keywords. Let W^i denote user i's profile, and let element $w^i_{jk} \in \mathcal{R}$ of array W^i represent the confidence to user j's evaluations on items including keyword k.

Information items are selected and provided to each user. For each item, the relevance for the user is calculated by using the following matching function.

$$match(W^i, \langle \boldsymbol{kw}_h, \boldsymbol{ev}_h \rangle) = \sum_{j}^{N_{user}} \sum_{k}^{N_{term}} w^i_{jk}\, ev_{hj}\, kw_{hk} \tag{3}$$

The system recommends items to the user in the order of relevance.

Given the information items from the system, the user inputs evaluation values for the items to the system. The system invokes the feedback facility after receiving these evaluations from the user. In the feedback phase of the collaborative filtering, the system records the evaluation information for later use. Nakif uses the document profiles to store the user's evaluations so that the system can modify the document profiles in the feedback phase. Nakif is an incremental filtering system, which means that it modifies user profiles to increase the performance of recommendations. Accordingly, the system modifies both document profiles and user profiles.

In the original Nakif, a user profile is modified by a feedback function. The feedback function is defined for a two-point scale that only uses *good* and *bad* (and ϵ). We redefine the feedback function to handle evaluations on a five-point scale as follows.

$$W^{i\prime} = W^i + ev_{hi}\ \delta\ \boldsymbol{ev}^t_h \boldsymbol{kw}_h \tag{4}$$

This function gives the confidence to evaluators that have evaluated an item similar to the user's on the keywords that the item includes.

2.3 Non-incremental Nakif

In this section, we present a framework and some methods for constructing a type of Nakif style content-based collaborative filtering without feedback. Collaborative filtering systems with feedback (incremental methods) depend on the time order of evaluations. If two users have evaluated an item, the later evaluator's evaluation is not reflected in the first evaluator's user profile. Additionally, even if the user's tastes change dynamically all at once, the old tastes still remain in the user's profile.

We are developing a non-incremental content-based collaborative filtering approach based on the Nakif method. Non-incremental methods have added advantages. For example, in incremental methods, the information used in the construction of profiles is bound by time. That is, incremental methods use only evaluation information given before the time of feedback. Non-incremental methods, in contrast, use all of the evaluations given between a time duration. This means that non-incremental methods can use future evaluations. They therefore can use more information to construct user profiles than incremental methods. Consequently, non-incremental methods offer the potential of a better performance than incremental methods.

Let $\mathcal{D}$ denote a set of information items. Each document $d_h \in \mathcal{D}$ is represented as a pair of a keyword vector $\boldsymbol{kw}_h$, which indicates the content of the item in $\{0, 1\}$ [1] and an evaluation vector $\boldsymbol{ev}_h$ which stores the evaluation given to the item. where an element of evaluation vector $ev_{hj} \in \{2, 1, 0, -1, -2, \epsilon\}$ is an evaluation value to item h by user j according to the given evaluation value on a five-point scale.

The element of user profile w^i_{jk} is regarded as the similarity of user i's evaluation with user j's evaluation on keyword k. Let D^i be a set of items that is evaluated by user i, i.e., $D^i = \{d_h \in \mathcal{D} | ev_{hi} \neq \epsilon\}$. Now, we define w^i_{jk} as follows.

$$w^i_{jk} = \frac{\sum_{d_h \in D^i} ev_{hi}\ ev_{hj}\ kw_{hk}}{||D^i \cap D^j||} \tag{5}$$

where $||x||$ represents the cardinality of the set x. User i's element of profile w^i_{jk} represents the rate of correspondence to user j concerning the evaluations of items with keyword k. If there is no common item with keyword k evaluated by both user i and user j, i.e., $D^i \cap D^j = \emptyset$, w^i_{jk} should then be determined in another way. Here, we use the value 0.0 as the default value because 0.0 is the mid point of user evaluations.

The profile constructing method above is very simple in that it only counts the number of evaluations. We also propose another case; where the correlations between users' evaluations should be used for constructing the user profile.

In the correlation method to construct the Nakif profile, confidence value w^i_{jk} (user i to user j on keyword k) is calculated as follows.

$$w^i_{jk} = \frac{\sum_{d_h \in D^i} (ev_{ih} - e\bar{v}_i)\ (ev_{jh} - e\bar{v}_j)\ kw_{hk}}{\sigma_i\ \sigma_j} \tag{6}$$

In the correlation method, the value ϵ is defined as the mean of the user because the mean is the most neutral value of evaluations.

3 Evaluation

In this section, we present a simulation-based experiment on a dataset of actual items and actual users. We used the MovieLens dataset for the simulation

[1] We are able to have a real value to a keyword vector depending upon the index strategy of the database.

and MAE and ROC as the measures. We examined the Pearson correlation coefficient method, a type of correlation-based collaborative filtering, for a comparison with the above Nakif methods.

3.1 Dataset

The MovieLens dataset is an evaluation dataset for collaborative filtering systems built by the GroupLens project [4].[2] It was collected throughout the MovieLens service, by a recommender system for movies implementing GroupLens technologies.[3]

The MovieLens dataset includes 100,000 evaluations for 1682 movies by 943 users. Each movie was evaluated on a five-point scale (bad 1 to good 5) and all of the users made at least 20 evaluations. Each movie was classified into several of 18 genres (19 including "unknown"). We used genres as the content information (keywords) in Nakif.

3.2 Measurement

The performance levels of the target methods were measured as follows. The evaluations taken from the MovieLens dataset were partitioned into 8 : 2 pairs named the *base set* and the *test set*. In the experiment, given the base set, the methods were measured to predict the evaluations by the users with the test set.

1. For each user, the basis of the condition of whether or not to provide items was determined, e.g., in Nakif, a user profile was constructed by repeating feedback and in the correlation-based method, a weight between users was calculated based on the relationship between their evaluation tendencies to items.

2. For items evaluated by the users in the test set, relevance values were calculated by using matching functions and sorted as prediction ranking.

3. Comparisons were made between recommendations and real evaluations to check whether predictions were appropriate or not.

A five-fold cross validation was used to obtain precise results [6]. Here, we prepared five different partitions of the dataset into training and test sets.[4] The experiment described above was executed on all five pairs of training and test sets and the results were averaged for the total result.

---

[2] `http://www.grouplens.org/data/`

[3] `http://movielens.umn.edu/`

[4] These partitions were generated by a program attached to the MovieLens dataset.

3.2.1 MAE

The mean absolute error (MAE) is used frequently in information filtering [1, 7]. For the predictions of N items $(p_1, \ldots, p_N)$ and the real evaluations of a user $(r_1, \ldots, r_N)$, error $E = (p_1 - r_1, \ldots, p_N - r_N)$ is calculated. Now, MAE $|\bar{E}| = (\sum_{i=1}^{N} |p_i - r_i|)/N$ is used to measure the error rates of the matching methods. In such uses, high performance filtering results in low MAE values. The standard deviation of MAE is also used, i.e., for the measurement of the stability of the filtering performance.

For a comparison with real evaluations, made on a five-point scale, we also created predication values on a five-point scale. Since the system can only output a ranked list of items in the order of relevance, we gave the predication values for each item to have the distribution of the real evaluation values in the test set according to the rankings.

3.2.2 ROC

The receive operating characteristic (ROC) is a measure for diagnostic systems and is used in experimental psychology [8]. With the ROC measure, we should restrict each recommendation as *good* or *bad* even if the system under study can only output a sorted list of items. We determine values 4 and 5 to be good like Herlocker et al. [4]. This is called the ROC-4 measure.

4 Results & Discussion

We used a correlation-based method for comparison purposes in the experiments. Correlation-based methods use similarity values between users as confidence values. For example, a similar tendency concerning an evaluation with another user implies a high confidence to the user.

We used the Pearson correlation coefficient, the incremental (original) Nakif, and a non-incremental version of Nakif in the evaluations. Table 1 shows results of the experiments on these three methods. In Nakif, we set $\delta = 0.01$ and the initial value of the user profile to 0.0. We can see that both Nakif and the non-

Table 1: Results

Methods	MAE	σ_{MAE}	ROC-4
Pearson correlation	.8138	.5251	.6188
Incremental Nakif	.7469	.4912	.6754
Non-incremental Nakif (simple)	**.7317**	**.4859**	.6780
Non-incremental Nakif (using correlation)	.7362	.4896	**.6806**

incremental Nakif mark higher performance levels than the Pearson correlation coefficient and that the non-incremental Nakif performs slightly more efficiently than the incremental version.[5]

In Nakif, several parameters should be determined for system execution. We analyzed the parameters in Nakif: the initial state of the user profile and the feedback factor δ. The user profile was initially created (1) randomly and (2) uniformly. The elements in the uniform profile all had values of 0.0 in the initial state. The values of δ were set to 0.1 and 0.01. The results in Table 2 show that we got a better performance when the difference δ was larger and the initial values of the user profiles were decided uniformly.

Table 2: Nakif results

	MAE		ROC-4	
	$\delta = .1$	$\delta = .01$	$\delta = .1$	$\delta = .01$
Random	.7447	.7511	.6710	.6638
Uniform	.7469	.7466	.6754	.6754
Non-incremental Nakif	**.7317**		**.6780**	

We also examined the system by precision/recall measurements (Figure 1). In the incremental Nakif, all elements in the initial user profiles were set to 0.0 and δ was set to 0.1 because these parameters had resulted in the highest performance levels in the above measurements.

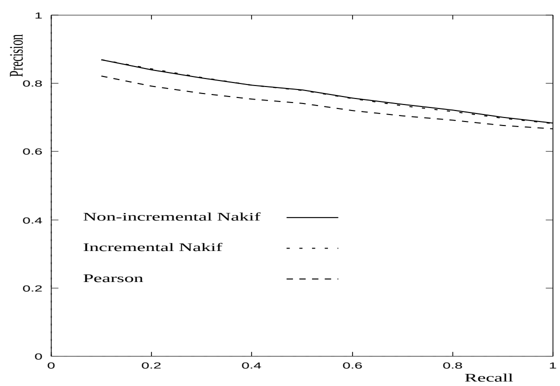


Figure 1: Precision/Recall graph

The results showed the incremental Nakif and non-incremental Nakif marking similar performances. We can therefore say that the non-incremental Nakif

[5]Herlocker's experiments resulted in MAE = .7982 and ROC-4 = .7127 for the normal Pearson correlation method and MAE = .7678 and ROC-4 = .7288 for the top-50 neighbor method [4].

marks a higher performance in total than the incremental Nakif, even though they are almost the same from the viewpoint of precision/recall.

Both the incremental Nakif and non-incremental Nakif performed better than the Pearson correlation coefficient method. The higher performance could be attributed to the content information used by Nakif, which is not used in the Pearson correlation coefficient.

5 Conclusion

In this paper, we presented experimental results from a content-based collaborative filtering method using a dataset of real users and real information items. The results showed that the incremental Nakif method marks a high performance for the dataset. Then, we introduced a non-incremental strategy. The non-incremental method had superior results than the incremental method although the non-incremental version used a very naive method. There is room for a more sophisticated method of constructing user profiles in the non-incremental Nakif framework.

The cold start problem still remains in collaborative filtering systems. We can use a non-incremental Nakif approach to make the initial user profile when there are available evaluation logs collected through recommender systems. The logging and usage of user evaluations is another issue of combining multiple collaborative filtering systems.

Although the non-incremental Nakif leads to better results than the incremental method in experiments, the non-incremental method has a disadvantage: it cannot efficiently recalculate user profiles for every feedback by using every evaluation given to the system in a sufficient time duration. Moreover, it requires that user profiles be modified to follow the changing interests and tastes of users. Therefore, it is desirable to use incremental modification for usual feedback and execute non-incremental profile reconstruction periodically and appropriately. The optimal combination of both methods remains as future work.

Acknowledgment

We thank the GroupLens Project, the University of Minnesota, for providing the MovieLens dataset.

References

[1] J. A. Delgado. *Agent-based information filtering and recommender systems on the internet.* PhD thesis, Nagoya Institute of Technology, 2000.

[2] K. Funakoshi and T. Ohguro. A content-based collaborative recommender system with detailed use of evaluations. In *Proc of 4th Int Conf*

on Knowledge-based Intelligent Systems & Allied Technologies (KES2000), pages 253–256, 2000.

[3] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communication of the ACM*, 35(12):61–70, 1992.

[4] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR'99*, 1999.

[5] J. Konstan, B. Miller, D. Maltz, J. Herlocker, L. Gordon, and J. Riedl. Applying collaborative filtering to usenet news. *Communication of the ACM*, 40(3):77–87, 1997.

[6] T. M. Mitchel. *Machine learning*. WCB/McGraw-Hill, 1997.

[7] U. Shardnand and P. Maes. Social information filtering: algorithms for automating "word of mouth". In *Proceedings of CHI'95*, pages 210–217. ACM, 1995.

[8] J. A. Swets. Measuring the accuracy of diagnostic systems. *Science*, 240(4857):1285–1293, 1988.

Who's Who in Recommender Systems

Workshop Attendees

Using Multiple Information Sources for Recommendation

Chumki Basu[†‡] Haym Hirsh[†]

{cbasu,hirsh}@cs.rutgers.edu

http://www.cs.rutgers.edu/~{cbasu, hirsh}

Most of the early recommender systems filter users who share similar tastes and hence are referred to as *collaborative filtering* systems [Hill *et al.*, 1995; Konstan *et al.*, 1997; Shardanand and Maes, 1995]. In some cases, the researchers' primary goal was to investigate the utility of recognizing communities of users based entirely on their preferences for a group of items, and not on analysis of the features of the items [Hill *et al.*, 1995]. However, content associated with the items plays an important role in *content-based filtering* systems [Pazzani and Billsus, 1997; Lang *et al.*, 1995], which often take a sample of the user's preferences and learn to predict what other items the user would like. Many of these systems use information retrieval based techniques in their data representation. In our work, we show content and collaborative information are different sources of information that can be integrated for use by recommender systems. We demonstrate this within the context of two different recommendation algorithms: inductive learning of user preferences and retrieval.

The motivations behind using multiple sources of information for recommendation can usually be linked to addressing the shortcomings resulting from using any single source in isolation. Consider the limitations of collaborative filtering systems when dealing with the various manifestations of ratings sparsity and poor ratings coverage. These include users who have provided few ratings, users who have few ratings in common with other users, and even items which have been rated by few users. It would make sense to use collaborative filtering when there are enough other users known to the system with overlapping characteristics. On the other hand, for content-based filtering, we usually make recommendations for items based on a single user's past preferences for similar data. Although we may be potentially limiting ourselves to an analysis of a narrower range of item choices, we are not dependent on having other users' ratings in order to make recommendations.

One of the early hybrid systems [Balabanovic and Shoham, 1997] addressed some of these concerns by using content to facilitate the comparison of users who may not have rated the same items in common. The system maintains content-based user profiles which are directly compared to find similar users for recommendation. An item is recommended to an individual user if it's found to be highly correlated with the user's profile or the profile of another similar user. Another early system which made recommendations and alternatively can be described as a personalized tour guide [Joachims *et al.*, 1997], also used elements of both collaborative and content-based approaches. In order to recommend Web pages, this system initially requests users to provide keywords describing their overall goals for the tour. As the user traverses links on the Web, the link descriptors are augmented with these keywords. This formed the basis for comparing the interests of different users along with the original textual description of the links. In other words, when a new user visits a Web page, his or her interests can be compared not only with a description of the link, but also with the interests of everyone else who has previously visited the link. Our own work on the inductive learning of movie preferences [Basu *et al.*, 1998] provides an example of how

[†] Department of Computer Science, Rutgers University, 110 Freylinghuysen Road, Piscataway, NJ 08854-8019.

[‡] Telcordia Technologies, Inc. 445 South Street, Morristown, NJ 07960-6438.

content-based information can be incorporated into the representation of the learning problem and learning can take place across a group of users, thereby using multiple sources of data for recommendation.

We have looked at two general approaches to recommendation, inductive learning and retrieval, both of which make use of content-based data. In our work on movie recommendation [Basu *et al.*, 1998], we consider recommendation as a classification problem. We are given a set of ratings for movies provided by a set of users. We would like to predict the users' preferences for new movies not yet seen. In particular, we rely on the Internet Movie Database, a semi-structured repository of movie data that is represented in a consistent format, as a source of content that can be used in the process of creating discrete features. We then learn a function that takes this data as input and that predicts whether or not a user will like a new movie. We differ from content-based methods in that other users' ratings data are processed by the learning system. This approach demonstrates that we are able to meaningfully utilize both ratings data and content in our problem representation. This approach is also flexible since we can incorporate content without making any modifications to the underlying recommendation algorithm.

In another approach to recommendation, we consider the problem of assigning conference paper submissions to suitable reviewers [Basu *et al.*, 2001]. We can view this as a variant of the general problem of technical paper recommendation. Current attempts to automate the conference reviewing process have converted it into a task that requires reviewers to rate keywords and sift through long lists of abstracts to find those that are appropriate for their interests. This is a laborious task for the reviewers. Relying on reviewers' review histories wouldn't alleviate the problem either, since this is sensitive information not easily maintainable from conference to conference. To ease the burden on the reviewers, we propose an automatic method for recommending small focused sets of papers to reviewers. Our approach is to retrieve for each reviewer a set of paper choices based on his or her interests. We construct representations of the reviewers using multiple sources of information that can describe their interests, namely their home pages and abstracts of their Postscript papers. Similarly, we use multiple information sources to describe the incoming papers, namely the papers' titles, keywords, and abstracts. Using information retrieval principles, we can compare these different representations to find papers which are close to the reviewers' interests, and generally find that using multiple sources helps achieve better performance.

In summary, we have found that content-based information plays an important role in the recommendation process. We have shown that we can integrate content directly into our problem representation to learn user preferences. We can also use content to represent user interests, and use this representation to retrieve related items from a separate collection of data. As part of our current work, we are also considering the use of demographics as another source of information. In general, we have found that using multiple sources of information can enhance the performance of our recommendation algorithms.

References

[Balabanovic and Shoham, 1997] M. Balabanovic and Y. Shoham. Content-Based, Collaborative Recommendation. In *Communications of the ACM.* 40(3). ACM Press.

[Basu *et al.*, 1998] C. Basu, H. Hirsh, and W. Cohen. Recommendation as Classification: Using Social and Content-Based Information in Recommendation. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, Madison, WI, 1998.

[Basu *et al.*, 2001] C. Basu, H. Hirsh, W. W. Cohen, C. Nevill-Manning. Technical Paper Recommendation: A Study in Combining Multiple Information Sources, *Journal of Artificial Intelligence Research*, Volume 14, p. 231-252.

[Hill *et al.*, 1995] W. Hill, L. Stead, M. Rosenstein, and G. Furnas. Recommending and Evaluating Choices in a Virtual Community of Use. In *Proceedings of the CHI-95 Conference*, 1995.

[Joachims *et al.*, 1997] T. Joachims, D. Freitag, and T. Mitchell. WebWatcher: A Tour Guide for the World Wide Web. *Proceedings of the Fifteenth International Conference on Artificial Intelligence.* Nagoya, Japan. 1997.

[Konstan *et al.*, 1997] J. A. Konstan, B. N. Miller, B. Maltz, J. L. Herlocker, L. R. Gordon, J. Riedl. GroupLens: Applying Collaborative Filtering to UseNet News. *Communications of the ACM.* Vol. 40, No. 3, March 1997.

[Lang *et al.*, 1995] K. Lang. NewsWeeder: Learning to Filter NetNews. *Machine Learning: Proceedings of the Twelfth International Conference.* Lake Tahoe, CA, 1995.

[Pazzani and Billsus, 1997] M. Pazzani and D. Billsus. Learning and Revising User Profiles: The identification of interesting web sites. In *Machine Learning* 27: 313-331.

[Shardanand and Maes, 1995] U. Shardanand and P. Maes. Social Information Filtering: Algorithms for Automating "Word of Mouth". In *Proceedings of the CHI-95 Conference*, 1995.

SkiMatcher™ Recommender System

Richard Davidson, MSc MBA. [richardd@Ski-Europe.com] CEO Ski Marketing Corporation dba SkiEurope (www.SkiEurope.com)

Background

SkiEurope is an established travel company that specializes in arranging European winter vacations and marketing them in North America.

The company's website (www.Ski-Europe.com) has existed since 1995, evolving from a pure information service toward e-commerce. The website has become the company's main source of individual bookings, a business segment that has grown substantially over recent years.

The current website objective is to turn visitors into buyers by encouraging them to request travel planning assistance or a trip price quotation.

We have hesitated to install an online booking engine since the winter vacation reservation process is complex (as compared to, say, air tickets). Moreover, studies have shown that buyers of complex travel products prefer personal interaction while making a purchase decision.

The Challenge

Research determined that many of SkiEurope's website visitors have made the decision to travel to Europe, but have not selected a specific destination.

To successfully sell our products, we found it important to staff our sales section with Europeans who have in-depth knowledge and experience related to the 60 destinations we offer. However, an inordinate amount of the staff's time was spent assisting customers to find the destination that best matched their requirements and preferences, before the booking process could begin.

From the inception of our website, we have sought a online means to move prospective customers along the decision-making continuum by empowering them to make their own better-informed more-confident destination selection prior to seeking offline assistance with their travel arrangement details.

The SkiMatcher™ Solution

In the spring of 2000, SkiEurope agreed to collaborate with TripleHop Technologies Inc. to develop the SkiMatcher™, a customized version of TripleHop's agent-based recommendation engines. This application uses next-generation artificial intelligence technology to help our website become a more personal and effective sales machine.

Like one of our experienced staff persons, SkiMatcher™ interacts with the user, learns their preferences, and responds with highly targeted, relevant information and personalized recommendations, supported by rich, original and customized content.

SkiMatcher™ includes a knowledge base of 60 European destinations, developed by more than 20 professional researchers to provide a comprehensive range of options and travel solutions for the winter vacation enthusiast. For each destination, the knowledge base comprises both quantified evaluation criteria and deep text content.

At the core of SkiMatcher™ is the Matching Engine™, TripleHop's cutting-edge, patent-pending, agent-based recommender system technology. By leveraging artificial intelligence, SkiMatcher™ captures and predicts users' interests through a combination of content filtering, collaborative filtering and click stream analysis. This sophisticated proprietary technology, has built-in learning mechanisms, so it understands more about our users with each interaction, generating ever more personalized recommendations.

The Results

SkiMatcher™ was beta launched in November 2000 to about 50,000 discerning subscribers to the SkiEurope Report. The response was overwhelmingly positive. The final version was released in December 2000.

By funneling users to the destinations, products or services that best match their preferences, we have increased our conversion rate significantly. In actual use, SkiMatcher™ users have shown a 70% higher conversion than non-users. Conversion is measured as the number of website visitors requesting planning assistance or a price proposal.

The Future

Currently, we are expanding and modifying the SkiMatcher™ to incorporate about 50 North American winter destinations. This comprehensive version has enabled us to conclude strategic partnerships with high traffic ski and travel websites for which solely European winter vacations were too narrow a niche to be viable. We also are implementing a HotelMatcher™ function.

In addition, we are working with TripleHop to apply our knowledge of SkiMatcher™ users' preferences to interact with our GoldMine™ CRM system in order to do data-mining and targeted one-to-one marketing efforts.

Collaborative Filtering and Item Clustering

Jan-Hendrik Dörner
Goethe-University, Frankfurt am Main (Germany)
doerner@wiwi.uni-frankfurt.de
http://www.ecommerce.wiwi.uni-frankfurt.de

1 Introduction

Since the beginning of Recommender Systems the main problems remain, although some solutions have been proposed: These problems seem (still) to be: How to

- handle the sparse data (especially at the start),
- make accurate predictions in real time, and
- measure the prediction quality.

The solution is generally believed to be in a model-based [2] approach in which additional data is used [1].

But receiving this data is more than complicated, especially when user enter regularly new non-text items to the system [5]. There is a trade off between the number of users using the system and the amount of available information on them (users do not like to enter personal data [6]).

2 Clustering in Theory

Clustering the item-space into groups of similar items can — theoretically — reduce the first two problems. By grouping similar items into homogeneous clusters, one can treat the whole cluster as a single point and therefore increase the number of overlapping votes. By grouping the items into clusters, one reduces the item space and is therefore increasing the speed in which predictions can be made. Last, but not least, there seems to be no good reason, why collaborative filtering is supposed to work outside clusters of similar items: Suppose two people agree on books on cooking, business, and comics, why should they have the same taste on novels?[1] Hence, only by clustering the item-space into homogeneous groups one can justify collaborative filtering to work. Those explanations might further be useful for acquiring the necessary trust in the system [4].

3 Hold–out Accuracy

Since one judges mainly items one knows, making predictions on a hold–out sample can lead to misleading results, since the hold–out is not at random, but taken from known items and is therefore more likely to be liked. This fact has to be taken into account when measuring the accuracy of Recommender Systems. On the other hand, one could also use this fact by interpreting a missing evaluation as an indication for a negative vote.

4 Empirical Evidence

The effect of clustering items for collaborative filtering is not clear, although [9] have shown that clustering has been useful on binary buying data from CDNow. In contrast [7] noticed that only in rare cases clustering items increases the prediction accuracy compared to the not partitioned case on movie rating data.

The results here have been calculated on the EachMovie–dataset, which has been gracefully provided by Digital Equipment, Inc. (now Compaq Computers). For clustering and evaluation only those movies and users were considered where each user has seen at least 15 movies and each movie was seen by at least 15 users. This restriction was made to eliminate results based on chance of too limited votes.

Two algorithms are compared:

- The **Pearson** prediction algorithm as described in [8], but improved with significance weighting of 50 (cf. [3]).
- The **Average** user prediction, which simply suggests the average of all remaining votes of the user.

To see, whether clusters might have any effect at all, the mean absolut error (MAE) between the prediction and the real vote (all but one) has been calculated in dependency of the number of votes of the user in total and in the cluster of the predicted item.

An ordinary least squares (OLS) estimation has been calculated to test, weather the prediction quality depends on the number of votes of the user

1. Dividing the item space according to content-based attributes could help at the beginning, but it might not be clear, what the appropriate attributes are. Are business-books really one big cluster? Do users differ between hard copies and paperbacks? Does the price matter? Questions like these can only be answered by looking at the user data, and the answer might not always be a priori clear.

in total or in the certain cluster. Table 1 show these results.

$$\text{MAE} = \text{Factor} \cdot \text{No. of Votes} + \text{constant}$$

	Pearson	Average
Number of user votes total	$3.881 \cdot 10^{-5}$ (0.584)	$6.371 \cdot 10^{-6}$ (0.108)
Number of user votes in cluster	$-5.208 \cdot 10^{-3}$ (8.611)	$3.960 \cdot 10^{-3}$ (7.340)

Table 1: Factor estimations

Interpreting the t-values (values in braces), the total number of votes has no significant effect, but the number of votes within the cluster. This first result encourages the use of clusters. A second regression tested, whether users rated more films in (non-trivial) clusters containing films they like.

$$\begin{aligned}\text{Vote} \quad = \quad & \text{constant} + \text{Factor} \cdot \\ & \text{Percentage of rated movies in cluster}\end{aligned}$$

The estimated factor was 0.523 (7.302), indicating that there is a significant positive tendency to watch and rate films one likes.

Finally, the dependency between the prediction accuracy (MAE) and the number of votes the user has in the cluster of the predicted film were calculated. These results are shown in table 2.

	Pearson	Average
Number of user votes in cluster ≤ 8	1.098	0.829
Number of user votes in cluster > 8	0.865	1.015

Table 2: Average prediction errors (MAE)

The results from table 2 show, that the (improved) Pearson prediction is only an improvement over the simple average user prediction, if there are several votes on similar items. This is in accordance with the theoretical thoughts of section 2.

5 Summary and Conclusion

The theoretical thoughts of clusters seem to hold up in practice: The accuracy of a prediction does not depend on the total number of votes of a user, but on the number on similar items. Only if there are enough votes on similar items, the Pearson prediction has a higher accuracy then the user average.

Users tend to rate movies they like. Therefore, holding back votes for measuring the accuracy might distort the real performance of collaborative filtering. Therefore, no rating might be interpreted as an indication for a dislike. Considering that many people rate only a few number of items, clustering might especially useful for those users with just a view votes: They should receive recommendations for items within the same cluster they rated in.

References

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

[2] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI–98)*, pages 43 – 52, Madison, WI, July 1998. Morgan Kaufmann Publishers.

[3] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings on the 22nd annual international ACM SIGIR conference on research and development in information retrieval*, pages 230 – 237, Berkeley, CA, August 1999.

[4] J. L. Herlocker, J. A. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. In *Proceedings of the ACM 2000 conference on Computer Supported Cooperative Work (CSCW)*, ACM SIGCHI Conference on Human Factors in Computing Systems, pages 241–250, Philadelphia, PA USA, December 2000.

[5] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *Proceedings of the International Joint Conference in Artificial Intelligence (IJCAI)*, pages 688 – 693, Stockholm, Schweden, July 1999.

[6] U. Manber, A. Patel, and J. Robinson. Experience with personalization on yahoo! *Communications of the ACM*, 43(8):35 – 39, August 2000.

[7] M. O'Connor and J. Herlocker. Clustering items for collaborative filtering. In *Proceedings on the 22nd annual international ACM SIGIR conference on research and development in information retrieval*, August 1999.

[8] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the ACM 1994 conference on Computer Supported Cooperative Work Conference (CSCW)*, pages 175 – 186, October 1994.

[9] L. H. Ungar and D. P. Foster. Clustering methods for collaborative filtering. In *Workshop on Recommendation Systems at the Fifteenth National Conference on Artificial Intelligence*, pages 114–129, Madison, WI, July 1998.

Recommendation in a Collaborative Digital Library Environment

Norbert Fuhr,* Norbert Gövert, Claus-Peter Klas
University of Dortmund, Germany

Federated digital libraries aim at covering most of a user's information needs. The DAFFODIL system[1] developed by our group is an agent-based architecture for accessing a relevant set of digital libraries in the area of computer science. DAFFODIL integrates searching and browsing, supports high-level search activities and allows for horizontal and vertical integration of services.

In order to support end users effectively, a digital library system should be integrated in the end user's work as much as possible. For this reason, we have started work for combining DAFFODIL with a groupware service that allows for storing digital library objects (DLOs), for annotating them and for sharing them with other users. Since we target at a large number of users, we want to include a recommendation service that suggests digital library objects to users or groups and that also supports the formation of user groups with similar interests.

In the following, we first briefly describe the collaborative digital library environment and then discuss recommendation issues.

DAFFODIL supports a number of DLOs such as bibliographic records as well as the fulltext of documents, references journals, conferences, authors and research groups. Each of these objects is assigned a unique identifier within DAFFODIL, which is achieved by means of a central object repository that stores descriptions of all objects currently referenced by any user of the system.

The groupware service allows a user to store DLOs in a hierarchy of folders. Users may form groups, either as closed project teams or open communities and share folders within these groups. (Communities and their folders are visible for users outside of this group, whereas project teams are not). Objects stored in the groupware service may be annotated by one or more users by attaching a discussion thread to them.

In this collaborative setting, the task of the recommendation service is to suggest DLOs to users or groups and to support the formation of new groups. Instead of relying on users' rating (empirical evidence from similar applications suggests that only little rating activity should be expected), the major basis for recommendation will be the organization of DLOs in folders. The general approach for recommendation is the comparison between users and/or groups based on their stored sets of DLOs. If there is sufficient overlap between the sets, the remaining objects can be recommended to the other users/groups not referencing these objects yet. In contrast to the standard recommendation procedure, however, there are two major differences:

1. Whereas standard applications consider one type of objects only, the data set forming the basis of the recommendation procedure consists of a variety of object

* fuhr@cs.uni-dortmund.de

[1] http://www.daffodil.de

types.

2. There are different kinds of relationships between DLOs, which should be taken into account when comparing users' profiles.
3. Usually, the data belonging to a single user forms a single set. In our setting, users organize their data in different folders, which are representing different contexts.

For the first problem, we envisage the assignment of different weighting factors for different object types (e.g. if two users reference the same article, this may not be as significant as referring to the same author). Vice versa, for the suggestion of new objects to a user, there may be different threshold values depending on the type of the object.

When comparing sets of DLOs, different kinds of similarity or relationships between objects should be considered:

- Similarity based on content: For documents, we usually have some metadata characterizing their content; for the other types of objects, however, the construction of appropriate content descriptions may be computationally expensive (e.g. by aggregating the descriptions of all documents published by an author or in a journal).
- Structural relationships between different object types: Here we mainly have author and aggregation relationships (e.g. document – journal, document – conference, author – research group). As a straightforward approach for considering these relationships in the recommendation procedure, related objects could be given a lower weight than identical objects.
- Relationships between objects of the same type. Most of these relationships are inferred from the author and aggregation relationships (e.g. co-authorship, co-occurrence of documents in a journal issue). In contrast, citation links are given explicitly. Again, different weighting factors could be used for considering these types of relationship.

The folder structure should be considered by the recommendation process, in order to distinguish between the different interests of a single user. Thus, instead of taking into account all DLOs referenced by a user, only objects within the same folder should be regarded for recommendation. However, since users may follow different strategies in organizing their folders, the size of a folder may vary considerably, thus making comparisons rather difficult. As a simple strategy, we plan to fix a minimum number of DLOs; if a folder contains fewer objects, the recommendation process will go up one level in the folder hierarchy and merge the contents of all folders below.

As mentioned before, the recommender service should also support the formation of groups, either recommending an existing community to a user or by suggesting two or more users to form a new group based on their common interests. Again the folder hierarchy should be considered for this task, since two users may work together even if they do not share all of their interests.

A major challenge in the development of the recommender system is the development of a scalable, distributed architecture. In order to allow for efficient interaction of users with their folders, the groupware services will be distributed such that a user has her private folder stored in a service located at her own institute, and group folders typically will be located at one of their members' institutes. On the other hand, the recommendation service should support a global view, in order to match users or groups located at different groupware services. Storing all relevant information in a single centralized database may not scale with a growing number of users. Thus, partitioning the database in an appropriate way will be necessary.

Oregon State University – Research in Intelligent Information Systems

http://www.cs.orst.edu/~herlock
http://www.nacse.org

1. INTRODUCTION

The Computer Science department at Oregon State University, together with the Northwest Alliance for Computational Science and Engineering (NACSE) has an extensive research program investigating how to design and build *intelligent information systems*. Information systems augment humans' ability to access and make use of information, while the "intelligent" indicates that the systems adapt to specific contexts of need and improve themselves over time.

We believe that recommender systems are a core component of intelligent information systems. As a result, we are involved in many recommender systems research projects. Our central goal is to develop intelligent information systems to a point where they are widely adopted and benefit as many people as possible. We also believe in dong research that produces software systems that are validated by large numbers of users. We describe some of the more relevant research projects

2. The Virtual Consultant

The Virtual Consultant is a intelligent information portal being designed by NACSE under contract from the Naval Oceanographic Office. The portal unifies a distributed set of web resources into a single portal, providing a consistent user interface, regardless of the location of the underlying document. We are developing a recommender system-based approach that tracks user clicks on links to identify relevance connections between queries and documents. When a user issues a new query, it is matched up with past queries and documents relevant to those past queries are recommended.

3. Context-aware Recommenders

Current recommendations systems collect users' rating of interest in items (or in categories of items). Given these ratings, a recommender system can predict the user's interest in general. However, such recommendations are independent of the user's immediate context of need.

In order to make recommender systems more generally useful, we need to build recommenders that are aware of the user's context of need. For example, a book recommender can tell you the five books that you are most likely to enjoy, but cannot answer the question "given that I need a book on Japanese cooking, what are the best recommendations?"

We are taking two approaches to making recommenders context aware. One is to integrate recommender systems with content-based retrieval methods such as relational databases and information retrieval engines. In such combined engines, a user supplies a query, and a recommender systems works together with the content-based ranking engine to recommend items that are both relevant and high-quality.

The second approach is to extend the concept of rating from just rating items to rating relationships. For example, we can have a user rate whether a document is similar to a query, whether two movies are similar, or whether a book is a good reference for a specified topic. In such a manner, we can provide content-aware recommendations even in domains where content-data is not available, which is common in non-textual domains.

4. Recommendation Algorithms

We are working hard to answer the question "what are the best algorithms and for what contexts?" Our focus is to develop algorithms that are both the most accurate and highly scalable. To this end, we are working closely with experts in the machine learning area. The primary challenges that we are faced with are sparsity of ratings and scalability.

5. Personalization & Privacy

Given that recommender systems rely on tracking usage in order to generate predictions, we are constantly faced with the issue of protecting privacy. Our group is currently investigating technological means that provide some level of privacy protection without significantly reducing the ability to personalize.

6. Graduate Studies in Intelligent Information Systems

If you are interested in pursuing a computer science graduate degree in intelligent information systems, consider joining the Intelligent Information Systems group at the Department of Computer Science at Oregon State University. We have many exciting projects, and we are growing. We offer both masters and PhD degrees. For more information, contact Jon Herlocker.

7. Principle Investigator & Contact

Jonathan L. Herlocker
herlock@cs.orst.edu
Oregon State University,
Department of Computer Science
102 Dearborn Hall, Corvallis, Oregon, US
541-737-8894

8. Associated Graduate Students

Anton Dragunov, Olivier Godde, Seikyung Jung, Xiaofeng Wang, Min Yang, Irwin Yoon

9. Associated Research Programmers

Tim Holt

Application of Recommender Systems to Mobile Phone Services

Keiichiro Hoashi* Takaya Tanaka* Shigeki Muramatsu* Naomi Inoue*
Yayoi Matsuoka** Masanobu Manei**

KDDI R&D Laboratories, Inc.*
2-1-15 Ohara Kamifukuoka, Saitama 356-8502 JAPAN
{hoashi, mura, takaya, inoue}@kddilabs.jp

Kyocera Communication Systems, Co., Ltd.**
3-5-24 Miyahara Yodogawa-ku, Osaka 523-0003 JAPAN
{yayoi-matsuoka, masanobu-manei}@kccs.co.jp

1. INTRODUCTION

Recently, many services based on mobile phones have become popular in Japan. Representative mobile phone services include downloading multimedia contents such as ring tones or wallpaper images. Users make access to such services to use downloaded data to customize their mobile phones. However, due to the rapid increase of content providers and the limited interface of the mobile phone itself, it is difficult for the typical user to search required contents by the mobile phone.

In this paper, we propose a system which, by using collaborative filtering techniques, automatically recommends contents to mobile phone users. The implementation of this system is expected to dramatically reduce the user's efforts to retrieve necessary contents by mobile phones.

2. EXISTING MOBILE PHONE SERVICES

The appearance of mobile Internet services such as EZweb and i-mode have caused a revolution in the way Japanese people use mobile phones. Prior to these services, mobile phones were used mainly for vocal communication, i.e., like normal telephones. Currently in Japan, over 42 million mobile phone users are able to access to the Internet by mobile phones, which is approximately 65% of all mobile phone users [1]. Services which are highly popular among users are e-mail and bulletin board services (BBS). There are also many sites which provide contents such as ring tones, and wallpaper image data. Users download such multimedia contents from these sites to customize their mobile phones.

Mobile Internet services currently provided by Japanese carriers generally begin at a portal site, which presents a list of representative mobile services. The initial list is usually a list of service categories, such as ring tones, pictures, news, etc. The user first selects a category from this list. As a result of selecting a category, a list of sites which provides the selected service appears on the user's phone, from which the user must select a site which satisfies their needs. The typical flow to download a ring tone from existing sites is illustrated in Figure 1.

Since the hierarchy of portal sites are decided by content providers, the user does not have preliminary knowledge about the structure of these sites. Therefore, the user is forced to roam around the site to find whatever he/she needs. This is extremely frustrating to the mobile phone user, since the interface is limited to a very small screen, and the user can only input search keys by using a small number of input keys.

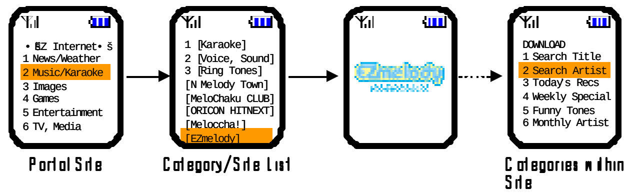


Figure 1. Typical flow of downloading ring tones by mobile phone

3. SYSTEM PROPOSAL

In order to reduce the user's frustration and improve system interface, we propose a new portal mobile phone site which automatically recommends contents to the user, and presents recommendations on the top page. Figure 2 illustrates the proposed system.

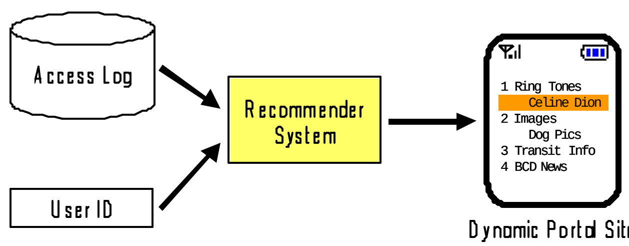


Figure 2. Proposal of dynamic portal site

In our proposed system, the recommender system makes recommendations based on access logs of each user and the log data of other users, and lists recommendations on the user's personal portal site. Recommendations are made periodically, so that the user's portal sites can be updated frequently. By providing such an interface, this system will reduce the interaction between the user and the mobile phone, which will result in a dramatic improvement of the interface of Internet access via mobile phones. Breese et al have made successful experiments on "implicit" vote data such as Web access logs. We believe the implementation of such collaborative filtering algorithms is suitable for our objectives.

4. REFERENCES

[1] Telecommunications Carriers Association: http://www.tca.or.jp/index-e.html

[2] Breese et al: Empirical analysis of predictive algorithms for collaborative filtering. Proc of 14th Conf of Uncertainty in Artificial Intelligence, pp 43-52, 1998.

Adaptive Humor Recommendation using Data Resampling

Jongwoo Lee Byoung-Tak Zhang

School of Computer Science and Engineering, Seoul National University

{jwlee, btzhang}@bi.snu.ac.kr

Abstrct

We constructed *MrHumor* System which analyzes the preferences of users on internet, combining collaborative filtering and contents-based filtering techniques, and recommends unrated humor documents to the users. As an adaptive learning system, *MrHumor* learns user's preference by selecting humor documents which maximize learning speed and users' satisfaction and querying documents' rating values to users and combines users' demographic profile, individual profile and common rating profile to operate stably without respect to the size and the class of data. We adopted PLSA (probabilistic latent semantic analysis) model as demographic profile learner, SVM (support vector machine) for individual profile learner, and adaptive k-NN model for collaborative filtering learner. *MrHumor* learns the models by resampling class boundary data for fast accuracy convergence and tackling the data sparseness problem , resulting in better estimation of document rating than ordinary learning method.

1. Contents

Information recommendation or information filtering technique filters out information of no interest and proactively recommends the information that is regarded as to be preferred by specific information consumers or users.

There are several methods such as *demographic profile method*, *individual profile method* and *collaborative method* in the presented data characteristic point of view. Demographic profile method and individual profile method are categorized as contents-based method as they learn profile from contents of the item while collaborative method is indifferent to the contents of item.

Demographic method constructs general demographic profile consisting of age, sex, hometown, majority, scholarship, vocation etc. related to the key features of item contents. This learned profile can be applied to the all users whose demographic profiles are known and recommendation can be performed, but users with the same demographic profile cannot be discriminated from each other consequently not showing the specific user's preference. But only this method gives the recommendation clue for the *new* users whose behavior to the item is unknown.

The drawback of demographic method can be overcome by learning the *individual* users' profiles. Each profile is learned from a user to represent the characteristic of an individual user using revealed user behavior to the item contents. But, these contents-based methods may show rather low performance if the key features of item contents are hard to grasp like the case of humor document. Collaborative method provides the alternative solution when learning the item contents is relatively hard. The basic idea of collaborative method is that a user showing the similar preference of the other in the past will probably show that of the other with a new item. But collaborative method suffers from the lack of information when the new item shows up and the rating information of other users on the new item is rare.

For recommendation system to work consistently well in cases of new user, new document and lack of common rating information, some kinds of combined model is needed. In our humor document recommendation system named *MrHumor* (mrhumor.snu.ac.kr) combines three methods above explained so that PLSA (probabilistic latent semantic analysis) model is used for demographic profile method, SVM (support vector machine) for individual profile method and adaptive k-NN for collaborative method. The combination method is primarily linear and varying with the rating data size based on the statistical optimization strategy.

In early recommendation stage, *MrHumor* largely works on the demographic profile method till individual users' preference information data and common rating data are accumulated. When the demographic profile method recommends, the recommended documents serve as *queries* which will be answered by the users. In this stage, we aim at both the prediction accuracy and fast model learning of the other two models. Since the data of rating document obtained by querying to the users tend not to be enough to learn the each component model in the system, rating data are *resampled* for the purpose of the fast convergence of learning model.

PLSA model for the demographic profile method learns the conditional distribution between the latent variables and the categorical indicator variables and also that between the latent variables and document's term vector. The distributions are iteratively updated by a kind of EM (expectation maximization) algorithm. Data used in this EM algorithm are resampled so that extremely characterized cases are more sampled than the other ones.

In *Mr. Humor* Resampling adopts QBC and partly a *bagging* (bootstrap aggregation) model.

The SVM model for the individual profile method is constructed by quadratic programming which actively finds the *support vectors* lying near the class boundary. When we resample the rating data, data near the mean of the rating values are more sampled than the extreme cases.

When the k -NN model of collaborative methods are utilized the k value is adaptively determined by the variance of rating values queried among users. The relationship between k value and the variance of rating values is obtained by linear regression method. Experiments will reveal the superiority of using resampled data to the normal case and the stability owing to combining the three models.

Personalizing with Recommendation Frames

Eliseo Reategui, Roberto Dias Torres Júnior
GetPersonal
Av. João Pessoa, 437/906, 90040-000 POA, RS, Brazil
{eliseo,roberto}@getpersonal.com.br

John A. Campbell
University College London
Gower St. WC1N 2BT, London, UK
jac@cs.ucl.ac.uk

Collaborative filtering and production rules are the most widely used techniques in recommender systems, each having their own advantages and disadvantages. We investigate the use of a still different approach, which we call *recommendation frames*. These *frames* are based on *category descriptors* (Reategui et al. 1997a) used in previous research to represent diagnostic knowledge. Both mechanisms are founded on ideas taken from a knowledge representation formalism called Knowledge Graphs (KGs). The graphs were obtained through a particular knowledge acquisition methodology which has been successfully applied in the acquisition of heuristic knowledge for medical diagnosis problems (Leão et al. 1990) and for the analysis of text understanding (Greco et al. 1987). In the context of recommender systems, a KG is a directed AND/OR acyclic graph with three types of nodes:

- *hypothesis nodes*: representing the possible outcomes, i.e. the items to be recommended;
- *evidence nodes*: representing other items, facts and user's features which influence the selection of different recommendations. They appear in the graph in their order of importance, from left to right;
- *intermediate nodes*: representing different groupings of evidence that when appearing together influence the selection of a recommendation.

The example presented in figure 1 shows that there are two different trees that lead to the recommendation. The first tree consists of the three leftmost nodes of the graph (A, B and C) connected by a logical AND in an intermediate node. The second tree consists of the fourth node of the graph (D), connected by a logical AND to the first tree. The two different trees can be described as two different alternatives to reach the recommendation, which means that they become connected by a logical OR.

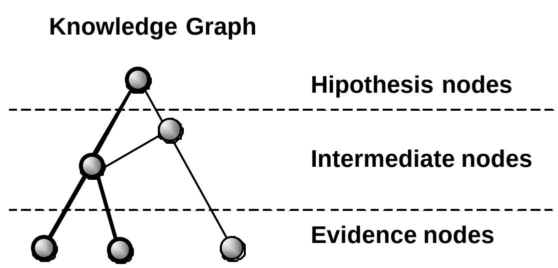


Figure 1: The basic structure of the knowledge graph

The knowledge graphs can be mapped into the *recommendation frames* using the strategy below:

- The leftmost node represents the most important *evidence* or the trigger for a given recommendation. This evidence has the feature of being specific for the recommendation;
- The leftmost tree groups a set of evidence that is observed frequently for the recommendation. They are *primary evidence* which can give a high degree of confidence for the outcome;

- The other evidence of the KG are *secondary evidence* which are not very frequent but which can reinforce the outcome (or determine it).

The *recommendation frames* can be modeled using the attributes *trigger, primary evidence* and *secondary evidence*, corresponding to the following elements of the KG, respectively: leftmost node, leftmost tree, remaining elements of the graph[1]. Figure 2 presents an example of this mapping.

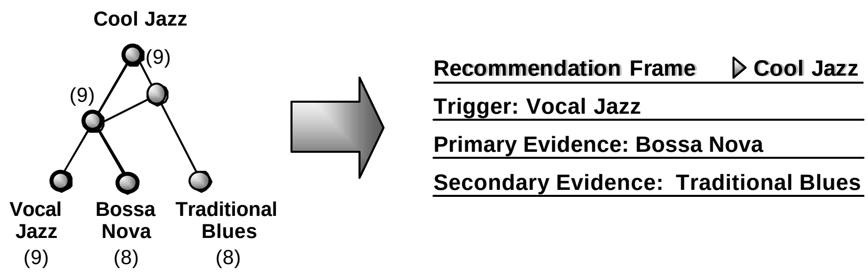


Figure 2: Mapping of the knowledge graphs into Recommendation Frame

In the example, the fact that a customer is interested in Vocal Jazz is a strong indicator that he will also like Cool Jazz. Being fond of Bossa Nova is another good sign of his preference for Cool Jazz, and showing interest in Traditional Blues reinforces the same premise.

The items described in the attributes *trigger, primary* and *secondary evidence* rank the most important evidence for each item that is suitable for recommendation, and they are learned through a particular method that is based on the combination of two distinct concepts: sensibility (the frequency with which a given piece of evidence appears in a class) and specificity (how often an evidence appears in one class, but not in others). The combined computation of the two concepts determine whether a fact will be part of the *triggers*, the *primary*, or the *secondary evidence*.

Previous research have shown a very good performance of a similar mechanism when solving diagnostic and classification problems (Reategui 1997b). Recent tests have shown us very promising results regarding the ability of the proposed technique in making accurate recommendations, providing explanations, and enabling manual fine-tuning of the knowledge stored in the *frames*.

References

Greco, G. & Rocha, A. F. (1987). The fuzzy logic of text understanding. *Fuzzy Sets and Systems*, 23: 347-360.

Leao, B. F. & Rocha, A. F. (1990). Proposed methodology for knowledge acquisition: a study on congenital heart disease diagnosis. *Methods of Information in Medicine*, 29: 30-40.

Reategui, E., Campbell, J. & Leao, B. F. (1997a). Combining a neural network with case-based reasoning in a diagnostic system. *Artificial Intelligence in Medicine*, 9(1): 5-27, January 1997.

Reategui, E., Campbell, J. A., Leao, B. F. (1997b). A case-based model that integrates specific and general knowledge in reasoning. *Applied Intelligence*, 7(1): 79-90, January 1997.

[1] To simplify the representation of the *recommendation frame*, the evidence corresponding to the *trigger* is not repeated in the list of *primary evidence* and other nodes.

The Role of Personified Agents in Recommendation Delivery

Eliseo Reategui
GetPersonal
Av. João Pessoa, 437/906, 90040-000 POA, RS, Brazil
eliseo@getpersonal.com.br

John A. Campbell
University College London
Gower St. WC1N 2BT, London, UK
jac@cs.ucl.ac.uk

1. Interfaces for recommendation delivery

Let us assume we have the best recommendation mechanism capable of retrieving, for a given customer, the most appropriate products from a large variety of choices. What would be the best way to present these recommendations? Recommendations may be presented in different ways in a web page. For instance, they may be intermingled with other products or occupy a limited zone of the page. According to the objective of the site one type of interface or another may be more appropriate:

- **Integrated recommendations:** In this type of interface one cannot easily distinguish whether the items contained in a page are actual recommendations or simply the contents of the page, which are displayed indistinctly to all users. While this type of interface has the advantage of being homogeneous and non-intrusive, it does not give the user the impression of being treated individually, which may be important if we are trying to work according to a one to one strategy. Furthermore, integrating recommendations in a web page indistinctly may violate a basic principle in a man-machine interface: it would prevent the user from feeling in control of the system as he would not know for sure where the recommendations are displayed.
- **Interrupting recommendations:** When we want to recommend a product to someone our urge to draw the person's attention, at any cost, is sometimes difficult to contain: pop-up windows, blinking banners, striking and flashing colours. This approach to recommendation delivery is based on interruption marketing, a strategy usually used by broadcast marketing to catch the public's attention in a world congested with advertising messages. It is true that we have to grab the customer's attention to the recommendation we are making. However, we should not turn our recommendations into disruptive messages. By interrupting the customers in intrusive ways we may lose their confidence and force them to develop mechanisms to simply ignore our messages.
- **Locale recommendations:** In this type of interface a small portion of the web page is pre-defined to contain the recommendations for the customer. This is what most e-commerce sites do. By delivering the recommendations in a pre-defined space we expect customers to always know where to find them. Besides, as the system gathers more information about the user, the page's recommendation spot may start being perceived as an added-service, a familiar space in the page where the user can identify himself, his interests and preferences.
- **Personified recommendations:** Personifying an interface means adding a human component to it, for example, by using a virtual character to communicate with and present recommendations to the user. The main advantage of this type of approach is that the use of the human figure can help us draw the user's attention, emphasising the recommendations made or highlighting any other message we may want to deliver. Furthermore, by centralizing the user's focus around the virtual character we may obtain an effect similar to that of interruption marketing, using a mechanism that is neither disruptive nor intrusive.

2. Discussion

Using the human figure in computer interfaces is still a controversial topic. Despite controversies recent research in the field of man-machine interface is starting to indicate that personification associated with personalization increases engagement and reduces workload [1]. De Angeli et al. [2] also present arguments supporting the use of personified agents. For instance, the authors mention that by introducing social stimuli, agents may improve the communication between user and computer. In a series of experiments we are starting to conduct we are also observing that users are more aware of recommendations delivered by the personified agent. The confidence the users sometimes build around the agent also makes them think that recommendations delivered by the them are more accurate. Some promising commercial results supporting the use of personified agents on the web are also starting to be shown. In a New York Times article of September last year the case of a virtual assistant who helps the user to buy flowers and bouquets via the internet is reported [3]. The assistant recommends different types of arrangements and flowers according to the customer's preferences. The flower shop registered an increase in sales of up to three times after the guidance of the virtual sales assistant. One of the reasons proposed for the success of the solution is that people want to buy from people, not from machines. But we may draw a slightly different conclusion: the human figure communicating with users captures their attention and makes them alert to messages delivered to them.

Although personality is a critical factor that may affect the acceptance of personified agents [4], adapting the agents to match the users' preferences can improve their communication with the users. A good signal showing that people are receptive to personified agents is that the market for such systems is growing very rapidly.

Providing explanations for the suggestion of a certain product is another way to improve the acceptance of recommendations. Researchers are investigating the best ways to build explanations with the main goal of giving the users more confidence in the suggestions that may be made. In Herlocker et al. [5] the authors affirm that the users are more likely to trust a recommendation when they know the reasons behind it. Their research group have also identified that explanations improve user involvement in the recommendation process, motivating them to add their knowledge as to improve the process of recommendation. The authors also observed an overwhelming request to see the explanation feature added to the recommender systems. Through the observation of users' comments, we have also identified that users are very keen to have recommendations explained. Adding explanation features to the personified agent's set of abilities would be another way to improve its effectiveness, ameliorate the communication with the user and improve the acceptance of recommendations.

References

[1] Jan Blom & Andrew Monk. One-to-one e-commerce. who's the one? In Proceedings of CHI 2001: Conference on Human Factors in Computing Systems. Seattle, Washington, 31 March - 5 April 2001.

[2] Antonella De Angeli, Paula Lynch, Graham Johnson. Personifying the e-market: A framework for social agents. To be presented at *Interact 2001 - Eighth IFIP TC.13 Conference on Human-Computer* Interaction, Tokyo, Japan, July 9-13, 2001.

[3] David J. Wallace. She's Only Code and Pixels, but She Can Help You Shop (http://www.nytimes.com/library/tech/00/09/biztech/technology/20wallace.html). The New York Times, September 20, 2000.

[4] Naas, C. & Lee, K.M. Does computer-generated manifest personality? An experimental test of similarity-attraction. In Proceedings CHI2000 – Conference on Human Factors in Computing Systems, ACM Press, pp. 49-57, 2001.

[5] Jonathan L. Herlocker, Joseph A. Konstan, John Riedl. Explaining Collaborative Filtering Recommendations. Computer Supported Cooperative Work 2000, Philadelphia, PA. December 2000.

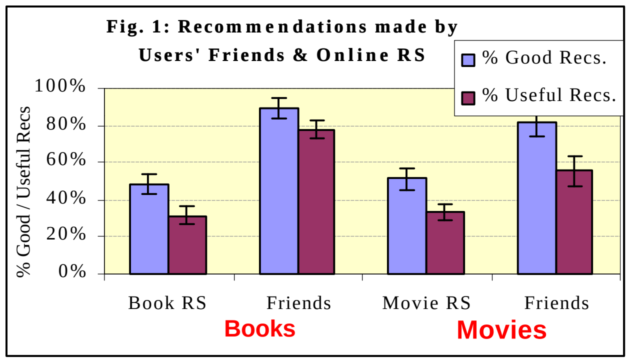
Benchmarking Recommender Systems

Rashmi Sinha & Kirsten Swearingen, SIMS, UC Berkeley, CA 94720
Email: sinha@sims.berkeley.edu; kirstens@sims.berkeley.edu
webpage: http://sims.berkeley.edu/~sinha/recommenders/html

A common way for people to decide what books to read / movies to watch is to ask their friends for recommendations. Online Recommender Systems (RS) attempt to create a technological proxy for this social filtering process. Since the goal of most RS is to replace (or at least augment) what is essentially a social process, we decided to directly compare the two ways of receiving recommendations (friends & online RS) from a user's perspective. Our second goal is to analyze users' interaction with online RS (input to the system, output from the system, and other interface factors). This will allow us to identify design features that go into making effective RS.

STUDY 1: In our first study (Sinha & Swearingen, Delos-NSF Workshop, 2001), we compared recommendations made by users' friends to those made by 3 book RS (Amazon, Sleeper and RatingZone) and 3 movie RS (Amazon, Reel and MovieCritic). We also evaluated the interface of the six RS. **Method:** 19 participants completed the following tasks for 3 book / movie systems: (a) Registered online, (b) Provided input to RS (i.e., rated items to receive recommendations), (c) Reviewed and evaluated output from RS, (i.e., rated the list of recommendations on several dimensions), (d) Completed satisfaction & usability questionnaires, and (e) Evaluated the recommendations provided by their friends (participants had previously given us e-mail addresses for 3 friends who could recommend 3 books or movies apiece). Participants evaluated recommendations from their friends along the same dimensions as those from online RS.

Measures for RS: Quality of Recommendations was evaluated using 3 metrics. (i) *Good Recommendations:* % of recommended items that the user liked. Good Recommendations can be divided into the following two subcategories. (ii) *Useful Recommendations* are "good" recommendations that the user has not experienced before. This is the sum total of useful information for the user—ideas for new books to read / movies to watch. (iii**)** *Previously Liked Recommendations (Trust-Generating Recommendations)* are "good" recommendations that the user has already experienced and enjoyed. These are not "useful" in the traditional sense, but our study shows that such items index a user's confidence in the RS.



Discussion of Results and Design Recommendations: Results showed that the users' friends provided a higher % of "good" and "useful" recommendations as compared to online RS (see Fig. 1). However, further analysis and post-test interviews revealed that users did like online RS. Users felt that recommendations from online RS were often "new" and "unexpected," while items recommended by friends mostly served as reminders of items users had already planned to pursue. Based on the results of the interface evaluation of the six online RS we offer the following observations and design recommendations for RS. (1) *Number of items users had to rate to receive recommendations:* Usability evaluation of the RS showed that users did not mind providing more input to the system in order to get better recommendations. (2) *Trust in System:* A surprising finding of our study was that Previously-liked recommendations play an important role in establishing trust in the system. Users' reasoning seemed to be: "If a system is recommending a lot of items I liked previously, I will probably like the new recommended items." (3) *System Transparency:* Another finding of our study was that users like to understand system logic, or "why was a particular item recommended." RS can achieve this by providing an explanation of why a recommendation was made. (4) *Level of Detail about Recommended Item*: Our results showed a link between liking for a recommended item and the amount of information provided by the system. Users found it difficult to judge a recommended item if the system provided few details. (5) *Interface matters, mostly when it gets in the way.* System navigation and layout seem to be the important factors (i.e., they correlate with ease of use and perceived usefulness of system). Color and graphics are less important.

FUTURE PLANS: One of the limitations of Study 1 is that users knew the source of recommendations, which recommendations came from systems and which from their friends. In our second study (currently underway) we have anonymized the source of recommendations to address this issue. This study focuses on several online music RS (e.g., Media Unbound, Music Buddha, SongExplorer, and Amazon). Music RS (unlike book and movie RS) allow users to experience recommended items immediately and provide feedback on the quality of the recommendations. We also plan to ask human music experts to provide personalized recommendations using the same input provided to online RS. Comparison between the systems' and the experts' performance constitutes a Turing test of sorts and should provide an interesting complement to the system vs. friends comparison.

TRIPLEHOP TECHNOLOGIES

Decision Support Tools for The Networked Economy

Who's who in Recommender Systems?
TripleHop Technologies, Inc.

Joaquin Delgado, PhD., CTO (Joaquin@triplehop.com)
1 World Trade Center, #5327, New York, NY 10048
Telephone: (212) 390-9090 x 222 Facsimile: (212) 390-1830
www.triplehop.com

Overview

We develop and market complex **decision support software** for corporate networks, intranets, extranets, e-commerce websites and call centers. All our solutions are based on our patent-pending matching technology, which uses **artificial intelligence** and is a proprietary combination of three levels of filtering (contextual filtering, *attribute-based collaborative filtering*, and click stream analysis). Based on this core technology, we have developed two series of applications in the fields of knowledge management and recommendations engines:

- **Knowledge Management Software**: According to Oracle Corporation, the amount of information flowing over corporate Intranets will be by 2006 two hundred times what it was in 1998. Other studies predict that the world is going to generate as much information and data in the next 3 years as has been generated since the creation of the universe. As the amount of information available grows exponentially every day, finding relevant data faster becomes mission critical. TripleHop's knowledge management software **MatchPoint**™, lets our clients retrieve information on the Internet, a corporate network and other sources of structured and unstructured data and efficiently share this information among a group of people. Our core technology enables the system to learn and become more "intelligent" with each new search.
- **Recommendation Software**: Getting customers from the information stage to the decision stage faster to increase sales is one of the most important challenges in today's environment. Our recommendation engines are turnkey, industry-specific software that can be implemented on a Web site in a matter of days. They include **TripMatcher**™, **ShopMatcher**™ and **JobMatcher**™. They are private label solutions that enable our clients to make personalized purchase recommendations to their users based on their profile, and convert them from "browsers into buyers." Testing on clients' Web sites has shown an increase of up to 70% in their conversion rate.

In just a few months of operations, we have developed two applications, signed up **twelve clients** and built a large sales pipeline. We have just signed up Orbitz**,** the industry consortium set up by American, Delta, Continental, Northwestern and United Airlines as well as JC Decaux, the world third largest outdoor advertisement agency. In addition to various partnerships, we have set up a crucial technology alliance with Artificial Life (Nasdaq: ALIF) that involves R&D efforts, the development of joint applications and a cross-distribution agreement.

Our **competitive advantage** is as follows:

1) We have a "pure technology" advantage. In particular (i) compared to rules-based engines, our technology continuously learns about users with each new interaction; (ii) compared to engines that only use collaborative filtering, we solve the problems typically associated with such technology, such as the cold start/new user problem, the lack of explanations and the portfolio effect;
2) We also have a "business" advantage, as we have positioned our products to better meet market needs. While solutions in our space are typically perceived as static, cumbersome, generic and difficult to implement, our products are adaptive, turnkey and industry-specific.

Our **Advisory Board**, which provides frequent guidance and contacts, includes:

- Michael J. Pazzani, Ph.D., is author of 2 books and over 100 technical papers on machine learning and intelligent agents, and is Chairman of Information and Computer Science Department at the **University of California, Irvine**. Michael is also the CEO of Adaptive Info, a provider of personalization tools for mobile portals.
- Jean Raby, an executive director at **Goldman, Sachs & Co**. with extensive experience of the software and technology industries.
- Andrew J. Malik is a Managing Director and has been with **Lehman Brothers** for fourteen years. His clients include Intel, IONA Technologies, Smart Force and Vitesse Semiconductor.
- Peter A. Derow is Vice-Chairman and Director of **EarthWeb**. He was previously President of CBS Publishing Group, Vice President and CBS, Inc. and Chairman and President of Newsweek and The Washington Post.

Product Mini-Descriptions

MatchPoint™

Collaborative Information Retrieval Solution

TripleHop Technologies is an innovative provider of information retrieval solutions for the access, integration and delivery of enterprise information. The MatchPoint solution collects, consolidates and analyzes information from multiple data sources from corporate documents to emails to Web pages, delivering to each unique user the information they require.

MatchPoint's predictive modeling technology and learning mechanisms make it - and you - more knowledgeable every day. This acquired knowledge is then distributed among all the employees who use MatchPoint, creating a virtuous circle within the organization.

Demonstrations of MatchPoint and details about the technology can be provided upon request.

TripMatcher™

How do you turn your lookers into bookers?

Imagine having a personal advisor on your site that understands, remembers and anticipates your visitors' travel preferences, and suggests to them the products that they are the most likely to buy. Introducing TripleHop's TripMatcher™ recommendation engine, a private label tool designed specifically for the travel industry. Now, with real-time profiling and text rating, you will get customer insight and automatically generate personalized and targeted purchase recommendations to your users. TripMatcher will reside on your Web site and will help turn your lookers into bookers by recommending the products, directly out of your database, that best match their needs. You can see TripMatcher in action at www.ski-europe.com ("Where to go?" section) and www.eurovacations.com (Destination Wizard).

About the Author: Joaquin Delgado - *Chief Technology Officer* - is a specialist of agent-based recommendation engines and information filtering on the Internet; he has published extensively on this topic and has been speaker at numerous AI related conferences and workshops. His main research interest is the blending of content and collaborative information filtering with probabilistic information retrieval techniques and machine learning. Prior to joining TripleHop he worked for at Chiyoda Corp. and Linc Media Inc. in Tokyo, Japan, handling and supervising complex Internet projects for companies pioneering in e-commerce. Prior to moving to Japan, Joaquin worked at Editora Ferga in Caracas, Venezuela, where he conceived and designed the first Venezuelan Export Directory Web site. He also worked at Oracle, where as an Oracle consultant, he developed extensive database design and implementation expertise. Joaquin holds a Ph.D. in computer science and artificial intelligence from Nagoya Institute of Technology, Japan, and a degree in computer engineering from Universidad Simon Bolivar, Caracas, Venezuela. He can be reached at (212) 390-9090 ext. 222 or by e-mail at Joaquin@TripleHop.com.

Summary of recommender systems related work at SRA division of IRST, Trento, Italy

The SRA division (Automated Reasoning Systems) at IRST (*http://sra.itc.it*) hosts a working group with a strong background in case-based reasoning (CBR). Recently our interests moved to the area of electronic commerce where the CBR approach appears very promising.

Recommendation systems overlap meaningfully with the CBR approach that relies on the assumption that similar users (problems) share similar solutions. From this perspective the similarity metric plays a key role in achieving effective recommender systems.

Recently we introduced the notion of compositional e-commerce that entails a revision of the standard framework of the recommender systems where given a user the goal is to provide a suggestion for a new good to buy. Our claim is that to recommend a set of goods differs from recommending one good at once many times.

Our research aims to address such a kind of problem starting from the CBR approach that seems to be a viable solution also for other typical problems of collaborative filtering. The drawback introduced by the user profiling is one example. We are looking at techniques that reduce the constraints posted by the needs of a user model.

Moreover we are trying to understand how to design recommender systems that exploit the preferences of the users formulated in terms of order relations. We believe that this perspective is very promising because it is much easier for the users to express their preferences in terms of relative choice rather than absolute ratings.

Projects:

CoCoA, Compilation Compiler Advisor

We developed an architecture combining CBR and Collaborative Filtering tecniques in the music domain. The task is to help the user in the process of assembling his own CD.

In the recommendation approach, the suggestion (the sound track) is provided without the context (the compilation) that could help the user to receive a sort of explanation of why the track proposed by the system has been selected. Furthermore, the customer cannot directly interpret and reuse what other customers have done in the past by looking at the existing compilations and deriving a new one from them.

Our claim is that to recommend a set of goods differs from recommending one good at once many times.

The first motivation of our work was to introduce the CBR loop in the context of compositional e-commerce, where a customer is required to collect a set of components to fulfill the pre-sales phase. Our thesis is that the cases are a viable solution to partition the recommended tracks into a meaningful collection to support the completion task.

A further motivation underlying our work is to integrate into the CBR loop a retrieval engine so that there is no need to address the content description in order to compute the similarity measure. In the domain of music this aspect is crucial for two reasons: firstly, because the definition of music categories does not meet a general agreement; and secondly, because the features that usually describe a sound track do not discriminate them between the different genres of use.

The online recommender is available at http://cocoa.itc.it/cocoakaradar.

CoCoA Radio, Combining Preferences for a Case-Based Radio Program

This ongoing project aims to build an on-line interactive hit parade radio. The idea is to give the radio listeners the possibility to influence the radio program by submitting their personal hit parade (ordered list of songs) relative to given music themes. The system will synthesize the new radio program trying to satisfy users' ordering preferences. Users are also invited to give feedback respect to the synthesized program. In this framework we are interested in learning how to effectively recommend ordered sequences of items.

Implicit Culture

Research on Implicit Culture aims at supporting new agents (artificial and humans) to act consistently with the knowledge and the behaviors of a group of agents. Implicit Culture is the relation existing between a set and a group of agents such that the elements of the set behave according to the culture of the group. Supporting Implicit Culture is effective in solving the problem of improving the performances of agents acting in an environment where more-skilled agents are active. A System for Implicit Culture Support, instead of working on the agents capabilities, modifies the view the agent has of the environment and, consequently, its actions with the objective to make it act consistently with the behaviour that a member of the group would have. Neither the new agent, nor a member of the group is required to know about it and so they share the same culture in an implicit way.

We have presented Implicit Culture as a generalization of Collaborative Filtering, in which the observed actions are voting and request for recommendation of item, the goal of the system is to make the agents acting coherently with the others and the environment is presented as recommended items.

Overhearing Agents for Recommender Agents

As part of our work on collaborative frameworks for agent-based systems, we introduced an abstract architecture (a pattern) based on a principle called Overhearing. The objective is to obtain an environment where unplanned collaboration in a community of artificial agents can be achieved by means of unobtrusive and unsolicited suggestions.

The architecture identifies three main roles, the "service" agents, the "overhearer" and the "suggester". In order to achieve their goals, service agents communicate by means of a public channel, observed by overhearers. Whenever a dialogue matches some given criteria, an overhearer notifies those suggesters that have declared an interest. The suggesters, depending on their own goals and the observed communication, may decide to send "suggestions" concerning the actions or objectives currently pursued by the service agents. Service agents are engineered to properly handle a set of (possibly contradictory) suggestions, in a context sensitive way (e.g., in order to adapt to specific needs or limitations of the HCI system in use).

The architecture sketched above is particularly suitable for recommender systems. It promotes a clear functional split between human interface and recommendation logic, software reuse and extensibility. Research on many aspects is still in progress, but it has already been adopted for a few experimental and commercial systems.

Researchers:

Stefano Aguzzoli[1]	stefano@gongolo.usr.dsi.unimi.it
Paolo Avesani	avesani@irst.itc.it
Enrico Blanzieri[2]	blanzier@psych.unito.it
Paolo Busetta	busetta@irst.itc.it
Antonia Doná	antodona@irst.itc.it
Paolo Giorgini[3]	pgiorgini@science.unitn.it
Paolo Massa	massa@irst.itc.it
Sabrina Recla	recla@irst.itc.it
Angelo Susi	susi@irst.itc.it

References:

S.Aguzzoli, P.Avesani, P.Massa. Compositional CBR via Collaborative Filtering International Conference on Case-Based Reasoning, Vancouver, July 2001.

F.Ricci, P.Avesani, Data Compression and Local Metrics for Nearest Neighbor Classification. IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 21, Number 4, April 1999.

P.Avesani, E.Blanzieri, F.Ricci. Advanced Metrics for Class-Driven Similarity Search. International Workshop on Similarity Search, Firenze, Italy, September 1999.

E. Blanzieri and P. Giorgini. From Collaborative Filtering to Implicit Culture, Proc. of the Workshop on Agents and Recommender Systems, in Agents2000, Barcellona 2000.

E. Blanzieri and P. Giorgini and P. Massa and S. Recla. Implicit Culture for Multi-agent Interaction Support. To appear in Proceedings of the Sixth International Conference on Cooperative Information Systems (CoopIS 2001), Trento, Italy, September 2001.

P.Busetta, L.Serafini, D.Singh, and F.Zini. Extending Multi-Agent Cooperation by Overhearing. To appear in Proceedings of the Sixth International Conference on Cooperative Information Systems (CoopIS 2001), Trento, Italy, September 2001.

M.Aiello, P.Busetta, A.Dona', and L.Serafini. Ontological Overhearing. To appear in Proceedings of the Eighth International Workshop on Agent Theories, Architectures, and Languages (ATAL 2001), Seattle, USA, August 2001.

Who's Who in Recommender Systems

Non-attendees

Amazon.com

"Our goal is to provide our 20 million users with 20 million different stores," Jeff Bezos explained at PC Expo 2000. "We're the most customer-centric company on earth."

Amazon.com personalization technology records what a shopper purchases, rates, or views on the website and then matches that data to the aggregated purchases of other shoppers who have bought similar products. By comparing what an individual shopper has purchased to aggregates of what shoppers with similar tastes have purchased, Amazon.com can provide highly targeted, relevant, and interesting recommendations to our customers. Features that use this technology on Amazon.com's website include New for You, Instant Recommendations, Gift Wizard, Recommendations Wizard, Similarity Explorer, Page You Made, "Customers who bought...", and personalized new releases.

"I want to transport online bookselling back to the days of the small bookseller, who got to know you very well and would say things like, 'I know you like John Irving, and guess what, here's this new author, I think he's a lot like John Irving.'", said Jeff Bezos. Of course, in Amazon.com's case, it's the technology, and not the antiquarian in the musty store, that remembers your likes and dislikes.

Amazon.com (www.amazon.com) opened its virtual doors on the World Wide Web in July 1995 and today offers Earth's Biggest Selection, along with online auctions and free electronic greeting cards. Amazon.com seeks to be the world's most customer-centric company, where customers can find and discover anything they might want to buy online. Amazon.com lists millions of unique items in categories such as electronics, kitchen and housewares, books, music, DVDs, videos, camera and photo items, toys, software, computer and video games, tools and hardware, outdoor living and wireless products. Amazon.com operates four international Web sites: www.amazon.fr, ww.amazon.co.uk, www.amazon.de and www.amazon.co.jp.

Recommender systems: Interfaces and Architectures

Marko Balabanović

flutter.com
175-179 St John St, London EC1V 4LW, UK
marko@bigfoot.com

INTRODUCTION

This paper provides a brief summary of research into interfaces and architectures for recommender systems, undertaken during 1994-98. A complete description can be found in [3].

We consider specifically the text recommendation task: helping the user follow threads of interest by regularly recommending small sets of 6-10 documents. Systems select a set of documents to present to the user, based on an induced user model; elicit feedback from the user (either explicit or implicit); and finally improve the user model given the feedback on the presented documents.

RESEARCH ISSUES

Two specific research issues are addressed by the research described here.

First, how to exploit the overlap between users' interests. In a system with multiple users, each with multiple topics of interest, resources can be allocated to serve areas where different users' interests overlap. Users' feedback can also be pooled—a collaborative strategy that can improve recommendations as well as reducing duplicated requests for user feedback.

Second, how to decide upon the composition of recommendation sets. Techniques for choosing individual articles for recommendation are well known, but another important consideration is how to compose a set of such recommendations (e.g., the selection of articles for an edition of a personalized newspaper).

COMBINING CONTENT-BASED AND COLLABORATIVE RECOMMENDATION

Content-based recommendation requires analysis of the content of items. This poses several problems: only certain domains can be readily analyzed (text is much easier than music or restaurants); systems will often over-specialize, only recommending items similar to those already rated highly. Collaborative recommendation instead computes similarity between users. As pointed out in [1], it avoids the problems of a pure content-based system. However it has its own shortcomings: new items cannot be recommended until some user has rated them, making very dynamic domains difficult; users with unusual tastes will be served poorly; negative feedback about items will not guarantee similar items won't be recommended again.

The *Fab* system combined content-based and collaborative recommendation to eliminate disadvantages of using either approach alone. A pool of *collection agents* adaptively gathered Web pages matching topic profiles. Each user's *selection agent* learned a personal user model and composed recommendation sets from the discoveries of the various collection agents and from other selection agents with similar models. The collection agents thus converged on areas of overlap between the users' interests, and the selection agents served both content-based and collaborative recommendations.

EXPLORATION vs. EXPLOITATION

A large-scale simulation was constructed in order to study the effects of varying the "breadth" of a set of recommendations [2]: whether to recommend items similar to those that have already received positive feedback, or new items where the user's reaction cannot be predicted. Experiments showed that an exploratory strategy increased the rate of learning user profiles, and the rate of adaptation to changing user interests, at the expense of showing users documents that are less often to do with their preferred topics. The result of this research was a new affordance for the Fab system allowing users to navigate among "broad" and "narrow" recommendation sets.

INTERFACES THAT CAPTURE IMPLICIT FEEDBACK

The *Slider* system [4] took these ideas further. Overlapped, sliding panels allowed users to specify multiple topics of interest and interactively alter the proportions between them. In addition, the interface did not require explicit feedback from a user, but successfully learned user profiles by monitoring drag-and-drop, selection and deletion actions. An "other news" topic was always available, showing items with the least overlap to those previously recommended.

REFERENCES

1. Balabanović, M., and Shoham, Y. Fab: Content-based, collaborative recommendation. *Commun. ACM 40*, 3 (1997).
2. Balabanović, M. Exploring vs. exploiting when learning user models for text recommendation. *User Modeling and User-Adapted Interaction 8*, 1 (1998).
3. Balabanović, M. Learning to Surf: Multiagent Systems for Adaptive Web Page Recommendation. Stanford University Department of Computer Science, PhD Thesis (1998)
4. Balabanović, M. An Interface for Learning Multi-topic User Profiles from Implicit Feedback. *Proc. of AAAI Workshop on Recommender Systems* (Madison WI, 1998), AAAI Press.

Research in Recommender Systems at Worcester Polytechnic Institute

Lee Becker, David Brown, Mark Claypool and Carolina Ruiz

http://www.cs.wpi.edu/Research/refer/

REsearch in FiltEring and Recommender systems (REFER) is a research group in the Computer Science department at Worcester Polytechnic Institute, in Worcester, Massachusetts, USA. The members of REFER share interests in issues relevant to the design and analysis of personalized information systems that recommend items of potential interest to their users on the basis of descriptions of these items ("content") as well as social ("collaborative") information about the relations between different users' tastes.

To provide personalized recommendations a recommender system must have ratings on each item from each user. The common solution, *explicit ratings*, where users tell the system what they think about a piece of information, is well-understood. However, stopping to enter explicit ratings can alter normal patterns of browsing and reading. Using *implicit ratings*, obtained by a method other than obtaining it directly from the user, has obvious advantages, including removing the burden on the user, and that every user-system interaction can contribute to an implicit rating. A main objective of our research is to collect, measure and evaluate the predictive power of implicit interest indicators, as begun in [CLWB01]. We are in the process of evaluating a large number of indicators, such as visiting a page, bookmarking, mouse movement, printing and time spent scrolling, and combinations of these indicators in order to determine which can be used to reliably predict interest in a Web page, and which might be combined. Through experiments, user studies and data analysis, we intend to show how implicit interest indicators can predict explicit interest, for all users, for groups of users, for an individual user, by using both individual and grouped implicit indicators.

We are also investigating the use of association rule mining as an underlying technology for collaborative recommender systems, as in [LAR02]. Association rules have been used with success in other domains. However, most currently existing association rule mining algorithms were designed with market basket analysis in mind. Such algorithms are inefficient for collaborative recommendation because they mine many rules that are not relevant to a given user. Also, it is necessary to specify the minimum support of the mined rules in advance, often leading to either too many or too few rules; this negatively impacts the performance of the overall system. We have developed a collaborative recommendation technique based on a new algorithm specifically designed to mine association rules for this purpose. This algorithm does not require the minimum support to be specified in advance. Rather, a target range is given for the number of rules, and the algorithm adjusts the minimum support for each user in order to obtain a ruleset whose size is in the desired range. Rules are mined for a specific target user, reducing the time required for the mining process. We employ associations between users as well as associations between items in making recommendations. Experimental evaluation of a system based on this algorithm reveals performance that is significantly better than that of traditional correlation-based approaches.

While collaborative recommendation combines the informed opinions of humans to make personalized, accurate predictions, content-based recommendation uses the speed of computers to make complete, fast predictions. Our research also explores new filtering approaches that combine the coverage and speed of content-based recommendation with the depth of collaborative recommendation, as in [CGM+99]. Our research also extends our earlier association rule approach [LAR02] to handle content-based recommendation, and a combination of content-based and collaborative recommendation.

[CGM+99] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes and M. Sartin, "Combining Content-Based and Collaborative Filters in an Online Newspaper", In *Proceedings of ACM SIGIR Workshop on Recommender Systems*, Berkeley, California, USA, August 1999

[CLWB01] M. Claypool, P. Le, M. Waseda and D. Brown. "Implicit Interest Indicators", In *Proceedings of ACM Intelligent User Interfaces Conference (IUI)*, Santa Fe, New Mexico, USA January 2001

[LAR02] W. Lin, S.A. Alvarez and C. Ruiz, "Efficient Adaptive–Support Association Rule Mining for Recommender Systems", *Data Mining and Knowledge Discovery Journal*, January 2002 (To Appear)

For the past year Britannica.com, Inc. has been conducting extensive analysis of the web usage logs from the Britannica.com website to improve various characteristics of the site. At this time, researchers at Britannica are hoping to take this analysis further to begin recommending encyclopedia articles, news stories, products, etc. to the users of the site based on past and current user behaviors.

Since this work is in the infantile stage, many aspects have not yet been defined. It is expected, though, that Britannica researchers will take into account user clicks and queries in an attempt to identify related concepts. These related concepts will be used to recommend web pages and products that may be of interest to a current user.

Researchers involved with this work:

Laura Mather
lmather@us.britannica.com

Kunal Sen
ksen@us.britannica.com

Britannica's website:
www.britannica.com

Recommender Systems at the University of Glasgow

Matthew Chalmers
www.dcs.gla.ac.uk/~matthew, matthew@dcs.gla.ac.uk

Our work spans theory and system development, and explores the use of a recommender system within multiple media and tools. It builds on earlier experimentation with a contextually-specific recommender for the Web and local file systems, *Recer* (Chalmers et al. 1998). Recer makes recommendations by comparing each person's ongoing context with his or her past activity and/or the past activities of others. Past activity here is represented by a time-stamped log of locations, documents, artefacts and people that each person has interacted with in all of the media we can: that person's *path*. Unlike most recommender systems, recommendations are specific to the ongoing context of the user, draw upon only the most relevant periods of past activity, and combine physical locations, hypermedia 'locations' and references, and locations in virtual environments. As discussed in (Chalmers 1999) and (Chalmers 2001), this approach is based on a combination of aspects of urban design theory, the trails of Bush's *As We May Think,* linguistics, semiology and philosophy of language.

Most of our work is centred in the *City* project. While we will gradually extend the range of places and topics it handles, the initial focal point is Charles Rennie Mackintosh, the architect, designer and artist. A rich body of 'people, places and things' related to Mackintosh exists here in Glasgow, including his reconstructed house within the Hunterian Gallery and Museum, the exhibition room devoted to his life and work within the Lighthouse Centre, and buildings he designed, e.g. the Art School and the Lighthouse itself. The Hunterian and the Lighthouse have agreed to be partners and test sites in this project, and in other related projects within the six year, eight site and £11,000,000 consortium called Equator. Equator (www.equator.ac.uk) is an 'interdisciplinary research collaboration' funded by the UK's Engineering & Physical Sciences Research Council and involving computer scientists, sociologists, psychologists, artists and designers. University College London, the Royal College of Art and the universities of Bristol, Glasgow, Nottingham and Southampton are taking part in the City project, with Glasgow as lead.

We are combining mobile computers, hypermedia and virtual environments in one system, and tracking each person's activity in them all as a path. We allow each person to interact with others even if they are using quite different media or combinations of media. We have found it useful to consider the many media, technologies and spaces as one design medium, because each person's experience depends on them all. People's activity continually combines and cuts across different media, interweaving those media and building up the patterns of association and use that make meaning. How people act and work is determined by the full combination of media that they can use and have used, and hence a narrow focus on only digital media as the paramount determinant of activity underrates the influence of other media. Irrespective of the media involved in logging and display, our algorithms for searching, matching and recommending do not rely on distinctions between these original media. They rely on their patterns of co-occurrence in human activity and hence in common semiological use. It is this that lets recommendations bridge across media usually held as separate in information systems, and lets us work with this everyday mix of types rather than against it. For example, a person who has been looking at the original chairs and tables made by Mackintosh may get recommendations of other physical artefacts to look at, but also of digital documents and virtual locations that might be of interest. These recommendations may come from people who have never seen the physical furniture, but have explored related digital information. One is offered information based on one's current location and the route one has recently taken, the information read and written, the artefacts one has shown an interest in, and how this activity relates to the activity of earlier readers, authors and visitors. The past routes and paths of curators, designers, authors and visitors are combined with current context to suggest recommendations for the immediate future.

We are creating a growing and evolving body of individuals' paths or narratives through the people, places and artefacts associated with the city of Glasgow. An initially static collection includes images, textual descriptions, and references to locations in the city, to artefacts in museums and exhibitions, and to electronic resources such as web pages and virtual 3D environments. Later, we will allow this body of information to grow as people use it, with their paths making new associations between symbols and also adding in new symbols. City includes information access based on objective bases, such as maps and search engines, and subjective bases, such as recommenders and synchronous audio links. We have now integrated VR and PDA access to the Mackintosh Room, will integrate a hypermedia interface within the next month, and the recommender within two months. We plan to demonstrate this prototype by October 2001, with synchronous and asynchronous awareness across all three media supported, paths interweaving symbols from all media and used as a resource for recommendations, and first experiences of non-Equator people to report and discuss. We have recently begun a series of semi-structured interviews with curators and exhibition designers in a number of UK museums, and a sociological study of related activity by city visitors and residents. These studies will feed reflection on our first full prototype and the design of future systems.

[Chalmers 1998] Chalmers, M., K. Rodden & D. Brodbeck: The Order of Things: Activity-Centred Information Access, In *Proceedings of the 7th World Wide Web Conference*, Brisbane.

[Chalmers 1999] Chalmers, M.: Informatics, Architecture and Language. In A. Munro, K. Höök & D. Benyon (eds.): *Social Navigation in Information Space*, Springer, 1999.

[Chalmers 2001] Chalmers, M.: Awareness, Representation and Interpretation, to appear in *J. Computer Supported Cooperative Work*.

Web Link and Traffic Analysis for Recommendation

Brian D. Davison
Department of Computer Science and Engineering
Lehigh University
davison@acm.org

The Web has two pervasive sources of activity that can be useful for recommendation systems. One is the traffic on the Web, reflected in individual user histories as well as proxy and origin server usage logs. The second is in hypertext link authoring. Both provide information that can often be treated as a value judgment. A user who clicks on a link, in effect, believes that there is some content of value at the destination. The significance of link creation is thus even stronger, as it reflects the judgment of the author (presumably) after viewing the target content and assessing it to be of sufficient value to spend the time to make the connection available to readers.

We have examined the relationship between the description of a page and its contents [4], and explored the value of link analysis for search engine result ranking and community discovery [8], as well as tested for nepotistic links that may influence such analysis [3].

Perhaps unlike most researchers in recommender systems, our interest is not only recommendation for human usage, but also recommendation for systems. Adaptive systems can then use those recommendations (from whatever source) to improve performance invisibly (unlike, for example Letizia [10] or WebWatcher [9] which alter the Web interface that is presented to the user). We are also concerned with appropriate evaluation of such systems.

While in the past we have examined techniques to build user models for user interfaces (such as the UNIX shell [7]), in recent years we have concentrated on issues related to modeling Web users for pre- loading of Web content to improve user- perceived performance [11]. We have proposed adaptive Web prefetching [1] which incorporates prediction from multiple sources, including:

- individual user histories,
- workgroup histories from proxy cache hints (a form of collaborative filtering),
- global user histories from origin server hints (again collaborative filtering),
- predictions from analysis of recently viewed content [6], and
- predictions from other available context, such as bookmarks, e- mail and usenet news.

Evaluation of Web pre- loading systems can be difficult. To that end, we have proposed [2] and implemented one approach to evaluate prefetching proxy caches simultaneously, and built and validated [5] a detailed network and caching simulator to estimate user latencies for testing and comparing alternate history- based prefetching approaches.

References

[1] B. D. Davison. Adaptive Web Prefetching. In *Proc. of the 2nd Workshop on Adaptive Systems and User Modeling on the WWW*, pp. 105–106, Toronto, May 1999.

[2] B. D. Davison. Simultaneous Proxy Evaluation. In *Proc. of the 4th Int'l Web Caching Workshop (WCW99)*, pp. 170–178, San Diego, Mar. 1999.

[3] B. D. Davison. Recognizing Nepotistic Links on the Web. In *Proc. of Workshop on Artificial Intelligence for Web Search*, pp. 23–28. AAAI Tech. Rep. WS- 00- 01, July 2000.

[4] B. D. Davison. Topical Locality in the Web. In *Proc. of the 23rd Annual Int'l Conf. on Research and Development in Information Retrieval (SIGIR 2000)*, Athens, July 2000.

[5] B. D. Davison. HTTP simulator validation using real measurements: A case study. In *Proc. of MASCOTS*, Cincinatti, OH, Aug. 2001.

[6] B. D. Davison. Predicting Web actions from HTML content. Under submission, 2001.

[7] B. D. Davison and H. Hirsh. Predicting sequences of user actions. In *Proc. of Workshop Predicting the Future: AI Approaches to Time-Series Problems*, pp. 5–12, Madison, WI, AAAI Tech. Report WS- 98- 07, July 1998.

[8] B. D. Davison, A. Gerasoulis, K. Kleisouris, Y. Lu, H. Seo, W. Wang, and B. Wu. DiscoWeb: Applying Link Analysis to Web Search. In Poster proc. of the 8th Int'l World Wide Web Conf., pp. 148–149, Toronto, May 1999.

[9] T. Joachims, D. Freitag, and T. Mitchell. WebWatcher: A Tour Guide for the World Wide Web. In *Proc. of the 15th Int'l Joint Conf. on Artificial Intelligence*, pp. 770–775. Morgan Kaufmann, Aug. 1997.

[10] H. Lieberman. Letizia: An Agent That Assists Web Browsing. In *Proc. of the Int'l Joint Conf. on Artificial Intelligence (IJCAI'95)*, pp. 924–929, Montreal, Aug. 1995.

[11] V. N. Padmanabhan and J. C. Mogul. Using predictive prefetching to improve World Wide Web latency. *Computer Communication Review*, 26(3):22–36, July 1996.

Eigentaste: A Constant Time Collaborative Filtering Algorithm

Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins
IEOR and EECS Departments
University of California, Berkeley
goldberg@ieor.berkeley.edu

Eigentaste is a collaborative filtering algorithm that uses *universal queries* to elicit real-valued user ratings on a common set of items and applies principal component analysis (PCA) to the resulting dense subset of the ratings matrix. PCA facilitates dimensionality reduction for offline clustering of users and rapid computation of recommendations. For a database of n users, standard nearest-neighbor techniques require $O(n)$ processing time to compute recommendations, whereas Eigentaste requires $O(1)$ (constant) time. We compare Eigentaste to alternative algorithms using data from *Jester*, an online joke recommending system.

Jester has collected approximately 2,500,000 ratings from 57,000 users. We use the Normalized Mean Absolute Error (NMAE) measure to compare performance of different algorithms. In the Appendix we use Uniform and Normal distribution models to derive analytic estimates of NMAE when predictions are random. On the Jester dataset, Eigentaste computes recommendations two orders of magnitude faster with no loss of accuracy.

(The Jester dataset including ratings from approximately 18,000 anonymous users is available by request: contact goldberg@ieor.berkeley.edu with contact information and a description of intended research.)

The joke recommending system *Jester* is available online at:
http://eigentaste.berkeley.edu/

The book recommending system *Sleeper* is available online at:

www.pmetrics.com

A .pdf version of our *Information Retrieval Journal* paper (July 2001) is available at:

www.ieor.berkeley.edu/~goldberg/pubs/

Sukumal Imudom
imudom@usc.edu
Computer Science Department
University of Southern California
University Park Campus
Los Angeles, CA 90089

Summary
I have been working on distributed annotation framework supporting collaborative filtering of any type of information. Currently, there exists the problem of validity and trust in recommendations placed on such systems. The framework provides a mechanism to build trust and reputation of the evaluators, as well as a reward model as an incentive for promoting evaluations.

The framework consists of three important building blocks aiming to overcome existing recommender system limitations: annotation semantic categories, expertise rating of the annotators, and a credit transfer mechanism. A generic set of annotation semantic categories is defined in order to help classify the free text annotations and provide annotation structure. The notion of "expertise" is introduced in the framework to help identify experts in our system. The expertise level of evaluators and a mechanism to identify the usefulness of evaluations can be used to help determine the credibility of the information. Additionally, we propose a credit transfer model to overcome the cold-start problem which is the lack of initial evaluations by providing incentives to promote user participation. This model, a hybrid of the subscription and transaction models, uses electronic credits as rewards for evaluators who make early and useful annotations.

The framework acts as a middleware allowing application developers to build various applications benefiting from the filtering process on top of it. The future use of these applications in the integration of recommender systems and Electronic Commerce is envisaged. For example, evaluators with credibility can earn community rewards. Such rewards can be mapped to money while giving their evaluations to customers. Not only would the future recommender systems help users more efficiently filter the quality of information, it also would provide a new way for consultants or analysts to do their jobs.

References
Sukumal Imudom and B. Clifford Neuman, **"A Framework Supporting Collaborative Filtering for Internet Information"**, AAAI-98 Workshop on Recommender Systems, Madison, Wisconsin, July 1998.

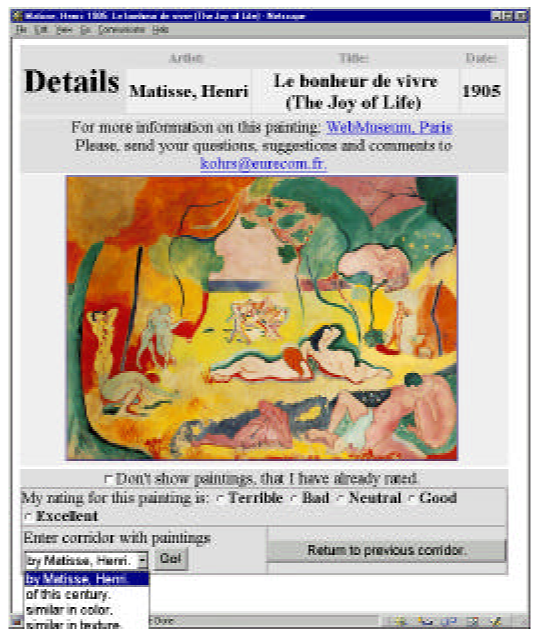
The Active WebMuseum

Arnd Kohrs, Prof. Bernard Mérialdo
Multimedia Communications Department
Institut EURECOM
BP 193, 06904 Sophia-Antipolis, FRANCE
{kohrs,merialdo}@eurecom.fr
http://bmgroup.eurecom.fr/

Project Summary

Our global interest lies in Information Filtering. In the Active Web Museum project, we are studying several issues related to the usage of collaborative filtering techniques :

- How to cope with the problem of insufficient data, in order to get reasonable performance even when little data is available,
- How to mix collaborative and content-based prediction,
- How to use prediction to personalize a web site.



The Active Web Museum is a web site which uses both collaborative and content-based techniques to predict paintings of interest for the user. Paintings are organized along virtual corridors, and the prediction value is used to order paintings inside the corridor. The user has to log on the site, then, if this is the first visit, he is asked to evaluate a small number of paintings, then he is free to visit the museum. During his visit, he can get a closer look at some paintings, rate the picture, or continue visiting other corridors. Different corridors are constructed to visit paintings by painter, by epoch, or by color and texture similarity.

The Active Web Museum is our web site, but we are also experimenting our methods on other available ratings databases. In particular, we are studying the automatic classification of users and objects, we have combined collaborative filtering with color and texture analysis of the painting. We have also proposed new evaluation schemes to measure the performance of the prediction in a user-oriented fashion.

References

[1] Arnd Kohrs and Bernard Merialdo.
Improving collaborative filtering with multimedia indexing techniques to create user-adapting web sites.
In Proceedings of the 7th ACM Multimedia Conference, Orlando. ACM, 1999.

[2] Arnd Kohrs and Bernard Merialdo.
Using category-based collaborative filtering in the Active WebMuseum.
In IEEE International Conference on Multimedia and Expo, New York, 2000.

Recommender Systems at the MITRE Corporation

Frank Linton
linton@mitre.org

The MITRE Corporation has developed three recommender systems for internal use. In contrast to commercial systems that recommend products to consumers, our systems recommend knowledge to individuals who are focused on performing their job tasks. One system, OWL, recommends new software functionality to individual users based on the functionality the individual's peers have found valuable. Another system, IURS, recommends URLs on the corporate Intranet to users; again based on the URLs the individual's peers have found valuable. The third system, ExpertFinder, locates individuals having some expertise, based on documents they have posted on the corporate Intranet.

In the OWL project (Linton & Schaefer, 2000) we instrumented an application (we instrumented Microsoft Word, but the method would work with any application) and gathered usage data from as many as 50 users at a time over a 3 year period. OWL compared the commands (by *commands* we mean cut, paste, file/open, file/print, etc) used by each individual to those used by the whole group and recommended new learning to each individual based on the difference between the individual’s and the group’s knowledge. The knowledge of which commands were worth knowing was constructed by the natural workplace activities of the group. It was not acquired by a conventional instructional design process nor transmitted by a human instructor. This instructional approach - monitoring a groups' behavior and generating individualized feedback - is now possible because monitoring is possible in the new networked workplace environment. The concept is not unique, OWL recommends learning in the same manner that commercial websites such as Amazon.com recommend purchases, but it is a radical departure from traditional instructional design, development, and delivery methods.

The MITRE Intranet contains over 800,000 technical and administrative documents. Despite the presence of links, search tools, keywords, and indexes, we suspect information is underutilized, resulting in duplication of effort. For this reason we have developed the Intranet URL Recommender System (IURS), a tool for information consumers. IURS determines which URLs - from a selected topic - one’s peers have visited earlier and guides the user to them. If the user identifies him- or herself to IURS, it will report only those URLs s/he has not already visited. Behind the IURS interface is a database of employee demographic and computer ownership data and a large dataset of Intranet server logs that is updated daily. These same data sources are also utilized by two other applications, one, on the Intranet portal, gives employees their current 'Top 10' URL recommendations, another enables Intranet information providers to see the demographics of their viewers.

The ExpertFinder system (Maybury, D'Amore, & House, forthcoming) helps workers find fellow employees who are knowledgeable about a particular subject area (such as "data mining" or "XML”). Individuals make their skills and knowledge known to the ExpertFinder (and therefore to others) by posting work products that utilize their skills and knowledge, typically documents, on the MITRE Intranet. ExpertFinder uses the existing search engine to find documents containing a user-specified keyword or phrase (e.g., "software agents") and uses a commercially available name extraction tool to find authors and names mentioned near the query terms within the documents. Each name is then weighted by several factors and the results are presented to the user in ranked order.

Recommender systems can improve workers' informal learning by collaborative bootstrapping. The main issues we have uncovered in this work are ensuring recommendation quality, developing reliable prototype software, maintaining data quality, and working in a fast-changing environment.

References

Linton, F., & Schaefer, H.P. (2000). Recommender Systems for Learning: Building User and Expert Models through Long-Term Observation of Application Use. User Modeling And User-Adapted Interaction: Special issue on Deployed User/Student Modeling Systems, 10, (2/3), 181-208.

Maybury, M., D'Amore, R. and House, D. forthcoming. Automated Discovery and Mapping of Expertise. In M. Ackerman, A. Cohen, V. Pipek, & V. Wulf (eds.). Beyond Knowledge Management: Sharing Expertise, Cambridge: MIT Press.

Recommender Systems Research at NTNU

Our research focus is on using agents to represent mobile customers and services [3] to perform service and product recommendations among a set of peer agents [1], in addition to applying agents as a general approach for targeted advertisements [2].

Research Institution

Department of Computer and Information Science (IDI),
Norwegian University of Science and Technology (NTNU)
IDI/NTNU, N-7491 Trondheim, Norway

People

Associate Professor	Mihhail Matskin
E-Mail	misha@idi.ntnu.no
Homepage URL	http://www.idi.ntnu.no/~misha/
Mobile	+47 9 189-7334
Dr. Matskin has works in agent technology, information systems, formal methods for software design and knowledge-based programming. His current research interests include telecommunication-based services, multi-agent system environments and agent-mediated e-commerce.	

Ph.D. Student:	Amund Tveit
E-Mail	amundt@idi.ntnu.no
Homepage URL	http://www.idi.ntnu.no/~amundt/
Mobile	+47 4 162-6572
Amund Tveit's research interests include agent-based software architecture, mobile commerce, recommender systems/collaborative filtering, information retrieval, web/data mining and mobile interface agents.	

Research Project

Name	**ElComAg -** The **El**ectronic **Com**mercial **Ag**ents Project
URLs	http://www.elcomag.com/ http://www.jfipa.org/

Publications

[1] A. Tveit. ”Peer-to-peer based Recommendations for Mobile Commerce”. *ACM Mobile Commerce Workshop*, Rome, Italy, July 21, 2001.

[2] M. Matskin. “Collaborative Advertising over Internet with Agents”. To appear in *Proceedings of the Intl. Workshop on Web Based Collaboration. WBC 2001*, Munich, Germany, September 3-7, 2001

[3] M. Matskin and A. Tveit. “Mobile Commerce Agents in WAP-Based Services”. *Special Issue on Mobile Commerce, Journal of Database Management*, IDEA Group Publishing, July-September 2001, pp 27-35

Stuart E. Middleton
IAM research group,
Department of Electronics and Computer Science,
University of Southampton, Southampton, S017 1BJ, UK
Email : sem99r@ecs.soton.ac.uk - Web : http://www.iam.ecs.soton.ac.uk/

Intelligence, agents and multimedia (IAM) group

One of the largest groups of its kind with over 80 researchers, IAM focuses on the design and application of computing systems for complex information and knowledge processing tasks.

IAM is a world leader in the key technologies of agent-based computing, knowledge management, open hypermedia and pervasive computing and their application in the domains of digital libraries and grids.

We see recommender systems as a way to bring together agent, user modelling and machine learning technologies, creating systems grounded in the real world that can provide useful service to real people.

Past Recommender System Work

The MEMOIR project

The MEMOIR system [1] was an open framework supporting a set of agents and hypermedia link services. User web browsing trails allowed MEMOIR to recommend colleagues with similar interests. MEMOIR was installed and evaluated by two end-user organizations.

Subsequent work moved from trails to bookmarks, introducing explicit annotation and replacing the underlying object-oriented database with agents implemented in Prolog.

The QuIC project

QuIC [2] explored the idea of dynamically adding multi-destination links to web pages based on the context of the pages. Content-based web page analysis was performed to create a context and this context used to recommend the most similar linkbase. Contextual links were added from the recommended linkbase to the users web pages.

The Quickstep project

The Quickstep recommender system [3] explored the acquisition of user profiles by unobtrusive monitoring of browsing behaviour and application of supervised machine-learning techniques.

Specifically, Quickstep addressed the problem of researchers finding relevant on-line research papers from the Internet. A detailed user profile was built from both unobtrusive monitoring of URL browsing and relevance feedback on recommendations provided. Content-based and collaborative filtering techniques were applied to formulate the daily recommendations.

Two identical experiments were conducted in order to evaluate its usefulness as a whole and test two profiling techniques. One profiling technique used a flat list of topics, the other an “is-a” ontological representation that allowed superclass interest to be inferred from more specific subclasses.

Both empirical studies suggested that the ontological approach was superior. A third, identical trial is planned to allow more rigorous statistical analysis to be applied.

Current Recommender System Work

The Foxtrot project

The Foxtrot recommender system is an evolution of the Quickstep system. It supports a searchable research paper database and offers recommendations based on observed behaviour and relevance feedback.

The major difference is that interest profiles can be visualized, so feedback can be provided directly about a profile. It is expected that a dialogue with the “profiling agent” will improve the quality of the feedback and enhance the users feeling of control over the recommendation process.

An experiment to measure the effectiveness of this enhanced feedback is to be conducted for 1 academic year, using about 150 undergraduates split into two groups. One group will be able to provide relevance feedback and direct profile feedback, the other just relevance feedback. The results will measure the comparative benefit direct feedback provides, and the overall effectiveness of such a recommender system.

Contact details

Stuart E. Middleton	sem99r@ecs.soton.ac.uk
Wendy Hall	wh@ecs.soton.ac.uk
Dave de Roure	dder@ecs.soton.ac.uk
Nigel R. Shadbolt	nrs@ecs.soton.ac.uk

REFERENCES

1. De Roure, D. Hall, W. Reich, S. Hill, G. Pikrakis, A. Stairmand, M. MEMOIR – an open framework for enhanced navigation of distributed information, Information Processing and Management, 37, 53-74, 2001
2. El-Beltagy, S. Hall, W. De Roure, D. Carr, L. Linking in Context. The Twelfth ACM Conference on Hypertext and Hypermedia (Hypertext '01), 2001, p.151-160
3. Middleton, S.E. De Roure, D. Shadbolt, N.R. Capturing knowledge of user preferences: ontologies in recommender systems, The First International Conference on Knowledge Capture (K-CAP-2001), Victoria, B.C.,Canada, Oct 2001

Multiagent Systems for the Social Competition Among Website Ratings

Matthias Nickles
Department of Informatics, Technical University of Munich
80290 München, Germany, nickles@cs.tum.edu

www7.in.tum.de/gruppen/kikog/projects/socionics/construct-s.htm

Summary

The self-description of websites and their hyper links referring to other sites usually reflect solely the requirements, opinions and preferences of the respective site owners, there is no explicit relation to the information, service and product offers of competing sites, and the assessment of the site by other users is unknown as well. To provide a "socially enhanced" solution for this problem, our project works towards open multiagent systems which (in contrast to usual filtering-oriented recommender systems) generate multi-dimensional, semantic website ratings which explicitly weigh and combine the possibly conflicting opinions of interacting rating agents which compete in the assertion of their individual ratings against others. These rating agents act as representatives of web surfers, interest communities, other web sites, private or public organizations, and represent their individual ratings in an open discussion forum which is attached to the rated web site. This forum is continuously observed by a rating instance which computes semantically rich, social weighted *general ratings* (of different abstraction levels) from the individual ratings as they show themselves in the agents communications. In contrast to the results of simple majority voting based recommender systems, a general rating makes social structures (like opinions, norms and roles) which emerge from communications *explicit* instead of filtering them regarding a certain user profile. In addition to the presentation of general ratings to the web surfer, these ratings can also be queried by external information agents (like web spiders) and by the rating agents themselves to improve their rating capabilities.
The described approach is based on the technical concept of the so-called *Social System Mirror* (SSM) component, which derives and models the social structures of a multiagent system and makes them visible both to the agents and to the application users. This concept is developed by the *ConStruct* project, which makes use of sociological theories to construct architectures for large adaptive multiagent systems ("artificial societies") modelled after human society. Take a look at our website for more information and related project publications.
Keywords: Semantic web, multiagent systems, collaborative filtering, socionics

Recommender Systems Research at NEC Research Institute

David M. Pennock, Steve Lawrence, Gary William Flake, Brian Whitman
NEC Research Institute, 4 Independence Way, Princeton, NJ 08540 USA
{dpennock,lawrence,flake,bwhitman}@research.nj.nec.com
http://www.neci.nec.com/

At NEC Research Institute, we are investigating new algorithms and applications for recommender systems. We have developed algorithms both for the purely collaborative case, and for combining collaborative and content-based data. We are actively pursing two major application areas: (1) recommending papers from ResearchIndex (`researchindex.org`), an online digital library of nearly 500,000 computer science papers [1], and (2) recommending music.

Our collaborators include Professor Lyle Ungar, Alexandrin Popescul, and Andrew I. Schein from the University of Pennsylvania; Professor John Riedl, Dan Cosley, and Shyong Lam from the University of Minnesota; Dr. Eric Horvitz from Microsoft Research; Adam Berenzweig and David Birchfield from Columbia University; and Professor C. Lee Giles from Pennsylvania State University.

Algorithms. Our *personality diagnosis* algorithm [3] is a pure collaborative filtering algorithm based on a probabilistic model of how people rate items. The algorithm empirically outperforms several previous methods on the EachMovie database of movie ratings, and on user profile data collected from ResearchIndex. The probabilistic framework naturally supports a variety of descriptive measurements—in particular, we consider the applicability of a value of information computation.

In other work [2], we examine some of the theoretical foundations of collaborative filtering algorithms, applying and extending axiomatizations from social choice theory.

Our generalized aspect models [4] offer a unified probabilistic framework for merging collaborative and content-based recommendations. The relative influence of collaboration data versus content data emerges naturally from the given data sources. Experiments on data from ResearchIndex show that appropriate mixture models incorporating secondary data produce significantly better quality recommenders than standard methods.

Recommender systems that rely solely on collaborative data fail when operating from a *cold start*—that is, when recommending items (e.g., first-run movies) that no member of the community has yet seen. In a paper appearing at this workshop [5], we develop several generative probabilistic models that circumvent the cold-start problem by merging multiple forms of content and collaborative data.

We plan to continue to pursue methods for combining content and collaborative data using generative probabilistic models. We are also exploring algorithms for using genre or other clustering information in recommender systems. Additionally, we are pursuing tools and techniques for scaling and generalizing recommendation algorithms to operate on massive and dynamic data sources.

Applications. We are developing a generic framework to allow anyone to evaluate recommendation algorithms using ResearchIndex. Developers can build on a skeleton client which handles communication with ResearchIndex and provides information about user actions and similarity between research papers. We also provide a fully-functional recommender engine using the personality diagnosis algorithm described above. We plan to evaluate recommenders using click-through rates, which provide a number of advantages over standard metrics such as mean absolute error. Standard metrics are often limited in domains where ratings are neither precise nor explicit, as is the case with ResearchIndex. Analyzing click-through rates can also provide an accurate picture of how a recommender impacts user behavior. Click-through is also a more relevant metric for the e-commerce applications typical of recommender systems.

We are also developing a music recommendation system, Minnowmatch, which combines collaborative and content-based recommendations. Minnowmatch uses analysis of the audio content of songs, as well as explicit and implicit user ratings.

References

[1] Steve Lawrence, C. Lee Giles, and Kurt Bollacker. Digital libraries and Autonomous Citation Indexing. *IEEE Computer*, 32(6):67–71, 1999.

[2] D. M. Pennock, E. Horvitz, and C. L. Giles. Social choice theory and recommender systems: Analysis of the axiomatic foundations of collaborative filtering. In *National Conference on Artificial Intelligence*, pages 729–734, 2000.

[3] D. M. Pennock, E. Horvitz, S. Lawrence, and C. L. Giles. Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach. In *Conference on Uncertainty in AI*, pages 473–480, 2000.

[4] A. Popescul, L.H. Ungar, D. M. Pennock, and S. Lawrence. Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In *Conference on Uncertainty in AI*, pages 437–444, 2001.

[5] A. I. Schein, A. Popescul, L.H. Ungar, and D. M. Pennock. Generative models for cold-start recommendations. In *Workshop on Recommender Systems at SIGIR*, 2001.

Altered Vista
http://alteredvista.usu.edu/

Researchers
Mimi Recker (mimi.recker@usu.edu)
Andrew Walker (andy.walker@usu.edu)
Department of Instructional Technology
Utah State University
2830 Old Main Hill
Logan, UT, 84332-2830, U.S.A.

Description
Altered Vista is a web based collaborative filtering tool that is geared toward educational web sites (both for students as well as teachers). In its current form, users explicitly submit a detailed rating form for web sites and then request recommendations that are generated using a neighborhood based correlation approach. The system has gone through several user trials and data for the most recent of these is available for download on the project site. Preliminary findings suggest that this particular user population tends to rate with a ceiling effect. While this results in extremely accurate predictions, these predictions do not outperform a simple non-personalized community average.

Support
This work reported is partially supported by a grant from the National Science Foundation (NSF NSDL DUE-0085855) and a Utah State University New Faculty Grant.

Instructional Architect
http://ia.usu.edu/

Researchers
Mimi Recker (mimi.recker@usu.edu)
Jim Dorward (jimd@coe.usu.edu)
David Wiley (wiley@cc.usu.edu)
Department of Instructional Technology
Utah State University
2830 Old Main Hill
Logan, UT, 84332-2830, U.S.A.

Description
Several initiatives have sparked the creation of digital libraries of on-line learning resources, also called 'learning objects'. While these collections contain some excellent resources, there is little provision for embedding these objects into new curriculum. Instructional Architect (IA) is an NSF funded project in development at Utah State University, which supplies a service layer on top of digital libraries. IA currently allows educational practitioners to search for learning objects and combine them into finished instruction. Work is underway to use collaborative filtering in recommending learning objects to users of the system based on the assumption that use of a learning object is an implicit vote of preference for that specific resource.

Support
This work reported is partially supported by a grant from the National Science Foundation (NSF NSDL DUE-0085855).

Phil Rennert
EC Wise Machine Learning Center
408 Saybrooke View Dr.,
Gaithersburg MD 20877
301-527-1230
prennert@ecwise.com

We do Web usability solutions, and we use a number of machine learning approaches, including collaborative filtering, to predict elements of page and site design which will be most useful and usable to a target audience. Predictions are based initially on demographics, later on user feedback and survey results; so we're yet another group grappling with the problem of how best to mix demographics and recommendations.

In a previous job, I wrote a collaborative filtering system to recommend colleges to high school students, based on the list of other colleges where they'd chosen to apply, for a Web-based college application service. From holdout test set results, the filter outperformed demographic clustering predictions and unpersonalized recommendations.

GroupLens Research: Recommender System Interfaces and Algorithms

PIs: John Riedl and Joseph A. Konstan
E-mail: {riedl, konstan}@cs.umn.edu
http://www.grouplens.org
4-192 EE/CS Building, University of Minnesota, Minneapolis, MN-55455

The GroupLens Research group has been working on Recommender Systems since we began exploring collaborative filtering in 1992. Our work has focused on two interlocking questions: what are the best interfaces for presenting recommender systems to users, and what are the best algorithms for producing recommendations. In this work we have run two major research systems: the original GroupLens for Usenet news (currently on hiatus), and MovieLens, a recommender for movies (www.movielens.org) with over 95,000 users and 6.7 million ratings.

In our research on interfaces we have recently been exploring new types of input from users that enable new ways to use recommenders, such as group recommenders, explanation systems, email-based recommenders, and ephemeral interest recommenders. We will briefly discuss the first two of these here; please write us to learn about the others.

Group recommenders make recommendations for items that are best for a whole group of people, though they may not be best for any one of the individuals. For instance, the group recommender for MovieLens can help a group of people find a movie to go to. The group recommendation is produced by combining individual recommendations with the goal of maximizing the "social good" for the group.

Explanation systems for recommenders provide an interface for users to understand better how it is that the recommender works, and why the recommender is suggesting a certain item. In MovieLens, our most common question is something along the lines of "Your recommendations are almost always exactly right. MovieLens really understands me. Therefore, how can you possibly think I would like *The Shining*?" The explanation can point out that people the user has agreed with about other movies like *The Shining*, can list other movies that are rated similarly to *The Shining*, or can otherwise explain what MovieLens is "thinking".

In our research on algorithms we have been exploring dimension reduction techniques, including singular value decomposition, we have studied the sensitivity of nearest neighbor recommenders to the most important parameters that influence their behavior, we have studied the metrics that are used to measure recommenders, and we have explored item-based recommenders. We will briefly discuss the first of those here.

Dimension reduction techniques have potential for recommenders both because they reduce the search space for finding good neighbors, or good items to recommend, and because they may help uncover latent relationships between the users or items that might be hard to discover in the high-dimensional space of the recommenders. Our research on singular value decomposition (SVD) has demonstrated that for some recommender tasks it offers much faster online performance, while producing reulsts that are as good as the best published results from other algorithms.

We are proud of our former Ph.D. students who are now actively pursuing recommender systems research in universities: Mark Claypool at Worcester Polytechnic Institute, Jon Herlocker at Oregon State University, and Ben Schafer at the University of Northern Iowa.

Dr. J. Ben Schafer
Department of Computer Science
University of Northern Iowa
Cedar Falls, IA 50614
Ben.Schafer@uni.edu

Prior Work

As a member of the GroupLens Research Project at the University of Minnesota I conducted research focusing on three aspects of recommender systems:

- A Taxonomy for Recommender Systems in Electronic Commerce

An analysis of how recommender systems have been applied in the field of electronic commerce was conducted. From this analysis, a taxomony for these systems was built, classifying them based on the functional I/O, the recommendation method, and other design issues such as degree of personalization and delivery method. Additionally, we identified six application models for recommender systems in electronic commerce. Finally, we conducted an extensive survey of electronic commerce sites and considered patterns related to which application models exist in which product domains.

- Filterbots

Information filtering agents (IF) and collaborative filtering (CF) both attempt to alleviate information overload by identifying which items a user will find worthwhile. Each of these techniques has advantages and limitations. This research demonstrated that the two can be beneficially combined. A CF framework was used to combine personal IF agents and the opinions of a community of users to produce better recommendations than either agents or user can produce alone. Additionally, it was concluded that a personal CF system consisting solely of a user's personal IF agents produced more accurate recommendations than individual agents alone or the recommendations produced by other combination techniques.

- Meta-recommender Systems

We have introduced a new class of recommender systems we classify as *meta-recommender systems*. These systems provide users with personalized control over the generation of a single recommendation list formed from the combination of rich recommendation data from multiple information sources and recommendation techniques.

As part of this research, we have presented the design and implementation of a meta-recommender framework named the *MetaLens Recommendation Framework* (MLRF). MLRF is based on an evaluation system model used by an increasing number of Internet sites to help consumers find items of interest. These sites allow consumers to identify a domain of purchase (i.e., a digital camera, computer, or automobile) and narrow the list of products within that domain. Consumers may indicate the features that they would be interested in, and the importance of these features in their final decision. Sites turn these requirements into search queries, use information filtering techniques over the attributes of known products in the category, and return ranked lists of "recommended" products. MLRF extends this model by evaluating how well individual consumers will *like* the items and including this evaluation in the recommendation process.

Research surrounding MLRF has focused on user preferences in meta-recommendation format, user perceived value of meta-recommenders, and usage analysis of three publicly available meta-recommenders built within MLRF for the domain of movies. Analysis of public use of these systems demonstrates that users like, and often prefer, these systems to more "traditional" recommenders. While acceptance comes at a slow pace, users who customized a system were more likely to return to use the system again. Finally, while the quantity and type of recommendation data preferred varies widely from user to user, analysis demonstrates that users want access to as much recommendation data as possible.

Current/Future Work

Having recently graduated from the University of Minnesota, I am currently in the process of establishing my research lab at the University of Northern Iowa. Research plans for the immediate future focus on interface and algorithmic issue for meta-recommenders. These include the creation of modular meta-recommenders, user-defined sources for recommendation data, interfaces for meta-recommendations through direct manipulation, and metrics for comparing the "accuracy" of different algorithms for meta-recommendations.