

Hybrid Recommender Systems: Survey and Experiments[†]

Robin Burke

California State University, Fullerton

Department of Information Systems and Decision Sciences

Keywords: recommender systems, collaborative filtering,
electronic commerce, case-based reasoning

Abstract

Recommender systems represent user preferences for the purpose of suggesting items to purchase or examine. They have become fundamental applications in electronic commerce and information access, providing suggestions that effectively prune large information spaces so that users are directed toward those items that best meet their needs and preferences. A variety of techniques have been proposed for performing recommendation, including content-based, collaborative, knowledge-based and other techniques. To improve performance, these methods have sometimes been combined in hybrid recommenders. This paper surveys the landscape of actual and possible hybrid recommenders, and introduces a novel hybrid, EntreeC, a system that combines knowledge-based recommendation and collaborative filtering to recommend restaurants. Further, we show that semantic ratings obtained from the knowledge-based part of the system enhance the effectiveness of collaborative filtering.

1. Introduction

Recommender systems were originally defined as ones in which “people provide recommendations as inputs, which the system then aggregates and directs to appropriate recipients” (Resnick & Varian 1997). The term now has a broader connotation, describing any system that produces individualized recommendations as output or has the effect of guiding the user in a personalized way to interesting or useful objects in a large space of possible options. Such systems have an obvious appeal in an environment where the amount of on-line information vastly outstrips any individual’s capability to survey it. Recommender systems are now an integral part of some e-commerce sites such as Amazon.com and CDNow (Schafer, Konstan & Riedl, 1999).

It is the criteria of “individualized” and “interesting and useful” that separate the recommender system from information retrieval systems or search engines. The semantics of a search engine are “matching”: the system is supposed to return all those items that match the query ranked by degree of match. Techniques such as relevance feedback enable a search engine to refine its representation of the user’s query, and represent a simple form of recommendation. The next-generation search engine Google¹ blurs this distinction, incorporating “authoritativeness” criteria into its ranking (defined recursively as the sum of the authoritativeness of pages linking to a given page) in order to return more useful results (Brin and Page, 1998).

One common thread in recommender systems research is the need to combine recommendation techniques to achieve peak performance. All of the known recommendation techniques have strengths and weaknesses, and many researchers have chosen to combine techniques in different ways. This article surveys the different recommendation techniques being researched — analyzing them in terms of the data that supports the recommendations and the algorithms that operate on that data — and examines the range of hybridization techniques that have been proposed. This analysis points to a number of possible hybrids that have yet to be explored. Finally, we discuss how adding a hybrid with collaborative filtering improved the performance of our knowledge-based recommender system Entree.

[†]To appear in *User Modeling and User-Adapted Interaction*.

¹ <URL <http://www.google.com/>>

In addition, we show that semantic ratings made available by the knowledge-based portion of the system provide an additional boost to the hybrid's performance.

1.1. Recommendation Techniques

Recommendation techniques have a number of possible classifications (Resnick & Varian 1997; Schafer, Konstan & Riedl 1999; Terveen & Hill, 2001). Of interest in this discussion is not the type of interface or the properties of the user's interaction with the recommender, but rather the sources of data on which recommendation is based and the use to which that data is put. Specifically, recommender systems have (i) background data, the information that the system has before the recommendation process begins, (ii) input data, the information that user must communicate to the system in order to generate a recommendation, and (iii) an algorithm that combines background and input data to arrive at its suggestions. On this basis, we can distinguish five different recommendation techniques as shown in Table I. Assume that I is the set of items over which recommendations might be made, U is the set of users whose preferences are known, u is the user for whom recommendations need to be generated, and i is some item for which we would like to predict u 's preference.

Collaborative recommendation is probably the most familiar, most widely implemented and most mature of the technologies. Collaborative recommender systems aggregate ratings or recommendations of objects, recognize commonalities between users on the basis of their ratings, and generate new recommendations based on inter-user comparisons. A typical user profile in a collaborative system consists of a vector of items and their ratings, continuously augmented as the user interacts with the system over time. Some systems used time-based discounting of ratings to account for drift in user interests (Billsus & Pazzani, 2000; Schwab, et al. 2001). In some cases, ratings may be binary (like/dislike) or real-valued indicating degree of preference. Some of the most important systems using this technique are GroupLens/NetPerceptions (Resnick et al. 1994), Ringo/Firefly (Shardanand & Maes, 1995), Tapestry (Goldberg et al. 1992) and Recommender (Hill et al. 1995). These systems can be either memory-based, comparing users against each other directly using correlation or other measures, or model-based, in which a model is derived from the historical rating data and used to make predictions (Breese et al. 1998). Model-based recommenders have used a variety of learning techniques including neural networks (Jennings & Higuchi, 1993), latent semantic indexing (Foltz, 1990), and Bayesian networks (Condliff, et al. 1999).

The greatest strength of collaborative techniques is that they are completely independent of any machine-readable representation of the objects being recommended, and work well for complex objects such as music and movies where variations in taste are responsible for much of the variation in preferences. Schafer, Konstan & Riedl (1999) call this "people-to-people correlation."

Demographic recommender systems aim to categorize the user based on personal attributes and make

Table I: Recommendation Techniques

Technique	Background	Input	Process
Collaborative	Ratings from U of items in I .	Ratings from u of items in I .	Identify users in U similar to u , and extrapolate from their ratings of i .
Content-based	Features of items in I	u 's ratings of items in I	Generate a classifier that fits u 's rating behavior and use it on i .
Demographic	Demographic information about U and their ratings of items in I .	Demographic information about u .	Identify users that are demographically similar to u , and extrapolate from their ratings of i .
Utility-based	Features of items in I .	A utility function over items in I that describes u 's preferences.	Apply the function to the items and determine i 's rank.
Knowledge-based	Features of items in I . Knowledge of how these items meet a user's needs.	A description of u 's needs or interests.	Infer a match between i and u 's need.

recommendations based on demographic classes. An early example of this kind of system was Grundy (Rich, 1979) that recommended books based on personal information gathered through an interactive dialogue. The users responses were matched against a library of manually assembled user stereotypes. Some more recent recommender systems have also taken this approach. Krulwich (1997), for example, uses demographic groups from marketing research to suggest a range of products and services. A short survey is used to gather the data for user categorization. In other systems, machine learning is used to arrive at a classifier based on demographic data (Pazzani 1999). The representation of demographic information in a user model can vary greatly. Rich's system used hand-crafted attributes with numeric confidence values. Pazzani's model uses Winnow to extract features from users' home pages that are predictive of liking certain restaurants. Demographic techniques form "people-to-people" correlations like collaborative ones, but use different data. The benefit of a demographic approach is that it may not require a history of user ratings of the type needed by collaborative and content-based techniques.

Content-based recommendation is an outgrowth and continuation of information filtering research (Belkin & Croft 1992). In a content-based system, the objects of interest are defined by their associated features. For example, text recommendation systems like the newsgroup filtering system NewsWeeder (Lang 1995) uses the words of their texts as features. A content-based recommender learns a profile of the user's interests based on the features present in objects the user has rated. Schafer, Konstan & Riedl call this "item-to-item correlation." The type of user profile derived by a content-based recommender depends on the learning method employed. Decision trees, neural nets, and vector-based representations have all been used. As in the collaborative case, content-based user profiles are long-term models and updated as more evidence about user preferences is observed.

Utility-based and *knowledge-based* recommenders do not attempt to build long-term generalizations about their users, but rather base their advice on an evaluation of the match between a user's need and the set of options available. Utility-based recommenders make suggestions based on a computation of the utility of each object for the user. Of course, the central problem is how to create a utility function for each user. Tête-à-Tête and the e-commerce site PersonaLogic² each have different techniques for arriving at a user-specific utility function and applying it to the objects under consideration (Guttman 1998). The user profile therefore is the utility function that the system has derived for the user, and the system employs constraint satisfaction techniques to locate the best match. The benefit of utility-based recommendation is that it can factor non-product attributes, such as vendor reliability and product availability, into the utility computation, making it possible for example to trade off price against delivery schedule for a user who has an immediate need.

Knowledge-based recommendation attempts to suggest objects based on inferences about a user's needs and preferences. In some sense, all recommendation techniques could be described as doing some kind of inference. Knowledge-based approaches are distinguished in that they have functional knowledge: they have knowledge about how a particular item meets a particular user need, and can therefore reason about the relationship between a need and a possible recommendation. The user profile can be any knowledge structure that supports this inference. In the simplest case, as in Google, it may simply be the query that the user has formulated. In others, it may be a more detailed representation of the user's needs (Towle & Quinn, 2000). The Entree system (described below) and several other recent systems (for example, [Schmitt & Bergmann, 1999]) employ techniques from case-based reasoning for knowledge-based recommendation. Schafer, Konstan & Riedl call knowledge-based recommendation the "Editor's choice" method.

The knowledge used by a knowledge-based recommender can also take many forms. Google uses information about the links between web pages to infer popularity and authoritative value (Brin and Page, 1998). Entree uses knowledge of cuisines to infer similarity between restaurants. Utility-based approaches calculate a utility value for objects to be recommended, and in principle, such calculations could be based on functional knowledge. However, existing systems do not use such inference, requiring users to do their own mapping between their needs and the features of products, either in the form of preference functions for each feature in the case of Tête-à-Tête or answers to a detailed questionnaire in the case of PersonaLogic.

2. Comparing Recommendation Techniques

All recommendation techniques have strengths and weaknesses discussed below and summarized in Table II. Perhaps the best known is the "ramp-up" problem (Konstan, et al. 1998). This term actually refers to two distinct but related problems.

² For example, see the college guides available at <<http://www.personalogic.aol.com/go/gradschools/>>

New User: Because recommendations follow from a comparison between the target user and other users based solely on the accumulation of ratings, a user with few ratings becomes difficult to categorize.

New Item: Similarly, a new item that has not had many ratings also cannot be easily recommended: the “new item” problem. This problem shows up in domains such as news articles where there is a constant stream of new items and each user only rates a few. It is also known as the “early rater” problem, since the first person to rate an item gets little benefit from doing so: such early ratings do not improve a user’s ability to match against others (Avery and Zeckhauser, 1997). This makes it necessary for recommender systems to provide other incentives to encourage users to provide ratings.

Collaborative recommender systems depend on overlap in ratings across users and have difficulty when the space of ratings is sparse: few users have rated the same items. The sparsity problem is somewhat reduced in model-based approaches, such as singular value decomposition (Strang, 1988), which can reduce the dimensionality of the space in which comparison takes place (Foltz, 1990; Rosenstein & Lochbaum, 2000). Still sparsity is a significant problem in domains such as news filtering, since there are many items available and, unless the user base is very large, the odds that another user will share a large number of rated items is small.

These three problems suggest that pure collaborative techniques are best suited to problems where the density of user interest is relatively high across a small and static universe of items. If the set of items changes too rapidly, old ratings will be of little value to new users who will not be able to have their ratings compared to those of the existing users. If the set of items is large and user interest thinly spread, then the probability of overlap with other users will be small.

Collaborative recommenders work best for a user who fits into a niche with many neighbors of similar taste. The technique does not work well for so-called “gray sheep” (Claypool, et al. 1999), who fall on a border between existing cliques of users. This is also a problem for demographic systems that attempt to categorize users on personal characteristics. On the other hand, demographic recommenders do not have the “new user” problem, because they do not require a list of ratings from the user. Instead they have the problem of gathering the requisite demographic information. With sensitivity to on-line privacy increasing, especially in electronic commerce contexts (USITIC, 1997), demographic recommenders are likely to remain rare: the data most predictive of user preference is likely to be information that users are reluctant to disclose.

Content-based techniques also have a start-up problem in that they must accumulate enough ratings to build a reliable classifier. Relative to collaborative filtering, content-based techniques also have the problem that they are limited by the features that are explicitly associated with the objects that they recommend. For example, content-based movie recommendation can only be based on written materials about a movie: actors’ names, plot summaries, etc. because the movie itself is opaque to the system. This puts these techniques at the mercy of the descriptive data available. Collaborative systems rely only on user ratings and can be used to recommend items without any descriptive data. Even in the presence of descriptive data, some experiments have found that collaborative recommender systems can be more accurate than content-based ones (Alspector, et al. 1997).

The great power of the collaborative approach relative to content-based ones is its cross-genre or “outside the box” recommendation ability. It may be that listeners who enjoy free jazz also enjoy avant-garde classical music, but a content-based recommender trained on the preferences of a free jazz aficionado would not be able to suggest items in the classical realm since none of the features (performers, instruments, repertoire) associated with items in the different categories would be shared. Only by looking outside the preferences of the individual can such suggestions be made.

Both content-based and collaborative techniques suffer from the “portfolio effect.” An ideal recommender would not suggest a stock that the user already owns or a movie she has already seen. The problem becomes quite tricky in domains such as news filtering, since stories that look quite similar to those already read may in fact present some new facts or new perspectives that would be valuable to the user. At the same time, many different presentations of the same wire-service story from different newspapers would not be useful. The DailyLearner system (Billsus & Pazzani, 2000) uses an upper bound of similarity in its content-based recommender to filter out news items too similar to those already seen by the user.

Utility-based and knowledge-based recommenders do not have ramp-up or sparsity problems, since they do not base their recommendations on accumulated statistical evidence. Utility-based techniques require that the system build a complete utility function across all features of the objects under consideration. One benefit of this approach is that it can incorporate many different factors that contribute to the value of a product, such as delivery schedule, warranty terms or conceivably the user’s existing portfolio, rather than just product-specific features. In addition, these non-product features may have extremely idiosyncratic utility: how soon something can be delivered may matter very much to a user facing a deadline. A utility-based framework thereby lets the user express all of the

considerations that need to go into a recommendation. For this reason, Guttman (1999) describes Tête-à-Tête as “product and merchant brokering” system rather than a recommender system. However, under the definition given above, Tête-à-Tête does fit since its main output is a recommendation (a top-ranked item) that is generated on a personalized basis.

The flexibility of utility-based systems is also to some degree a failing. The user must construct a complete preference function, and must therefore weigh the significance of each possible feature. Often this creates a significant burden of interaction. Tête-à-Tête uses a small number of “stereotype” preference functions to get the user started, but ultimately the user needs to look at, weigh, and select a preference function for each feature that describes an item of interest. This might be feasible for items with only a few characteristics, such as price, quality and delivery date, but not for more complex and subjective domains like movies or news articles. PersonaLogic does not require the user to input a utility function, but instead derives the function through an interactive questionnaire. While the complete explicit utility function might be a boon to some users, for example, technical users with specific purchasing requirements, it is likely to overwhelm a more casual user with a less-detailed knowledge. Large moves in the product space, for example, from “sports cars” to “family cars” require a complete re-tooling of the preference function, including everything from interior space to fuel economy. This makes a utility-based system less appropriate for the casual browser.

Knowledge-based recommender systems are prone to the drawback of all knowledge-based systems: the need for knowledge acquisition. There are three types of knowledge that are involved in such a system:

Catalog knowledge: Knowledge about the objects being recommended and their features. For example, the Entree recommender should know that “Thai” cuisine is a kind of “Asian” cuisine.

Functional knowledge: The system must be able to map between the user’s needs and the object that might satisfy those needs. For example, Entree knows that a need for a romantic dinner spot could be met by a restaurant that is “quiet with an ocean view.”

User knowledge: To provide good recommendations, the system must have some knowledge about the user. This might take the form of general demographic information or specific information about the need for which a recommendation is sought. Of these knowledge types, the last is the most challenging, as it is, in the worst case, an instance of the general user-modeling problem (Towle & Quinn, 2000).

Despite this drawback, knowledge-based recommendation has some beneficial characteristics. It is appropriate for casual exploration, because it demands less of the user than utility-based recommendation. It does not involve a start-up period during which its suggestions are low quality. A knowledge-based recommender cannot “discover” user niches, the way collaborative systems can. On the other hand, it can make recommendations as wide-ranging as its knowledge base allows.

Table II summarizes the five recommendation techniques that we have discussed here, pointing out the pros and cons of each. Collaborative and demographic techniques have the unique capacity to identify cross-genre niches and can entice users to jump outside of the familiar. Knowledge-based techniques can do the same but only if such associations have been identified ahead of time by the knowledge engineer.

All of the learning-based techniques (collaborative, content-based and demographic) suffer from the ramp-up problem in one form or another. The converse of this problem is the stability vs. plasticity problem for such learners. Once a user’s profile has been established in the system, it is difficult to change one’s preferences. A steak-eater who becomes a vegetarian will continue to get steakhouse recommendations from a content-based or collaborative recommender for some time, until newer ratings have the chance to tip the scales. Many adaptive systems include some sort of temporal discount to cause older ratings to have less influence, but they do so at the risk of losing information about interests that are long-term but sporadically exercised (Billsus & Pazzani, 2000; Schwab, et al. 2001). For example, a user might like to read about major earthquakes when they happen, but such occurrences are sufficiently rare that the ratings associated with last year’s earthquake are gone by the time the next big one hits. Knowledge- and utility-based recommenders respond to the user’s immediate need and do not need any kind of retraining when preferences change.

The ramp-up problem has the side-effect of excluding casual users from receiving the full benefits of collaborative and content-based recommendation. It is possible to do simple market-basket recommendation with minimal user input: Amazon.com’s “people who bought X also bought Y” but this mechanism has few of the advantages commonly associated with the collaborative filtering concept. The learning-based technologies work best for dedicated users who are willing to invest some time making their preferences known to the system. Utility- and knowledge-based systems have fewer problems in this regard because they do not rely on having historical data about a user’s preferences. Utility-based systems may present difficulties for casual users who might be unwilling to tailor a utility function simply to browse a catalog.

Table II: Tradeoffs between Recommendation Techniques

Technique	Pluses	Minuses
Collaborative filtering (CF)	A. Can identify cross-genre niches. B. Domain knowledge not needed. C. Adaptive: quality improves over time. D. Implicit feedback sufficient	I. New user ramp-up problem J. New item ramp-up problem K. “Gray sheep” problem L. Quality dependent on large historical data set. M. Stability vs. plasticity problem
Content-based (CN)	B, C, D	I, L, M
Demographic (DM)	A, B, C	I, K, L, M N. Must gather demographic information
Utility-based (UT)	E. No ramp-up required F. Sensitive to changes of preference G. Can include non-product features	O. User must input utility function P. Suggestion ability static (does not learn)
Knowledge-based (KB)	E, F, G H. Can map from user needs to products	P Q. Knowledge engineering required.

3. Hybrid Recommender Systems

Hybrid recommender systems combine two or more recommendation techniques to gain better performance with fewer of the drawbacks of any individual one. Most commonly, collaborative filtering is combined with some other technique in an attempt to avoid the ramp-up problem. Table III shows some of the combination methods that have been employed.

3.1. Weighted

A weighted hybrid recommender is one in which the score of a recommended item is computed from the results of all of the available recommendation techniques present in the system. For example, the simplest combined hybrid would be a linear combination of recommendation scores. The P-Tango system (Claypool et al. 1999) uses such a hybrid. It initially gives collaborative and content-based recommenders equal weight, but gradually adjusts the weighting as predictions about user ratings are confirmed or disconfirmed. Pazzani’s combination hybrid does not use numeric scores, but rather treats the output of each recommender (collaborative, content-based and demographic) as a set of votes, which are then combined in a consensus scheme (Pazzani, 1999).

The benefit of a weighted hybrid is that all of the system’s capabilities are brought to bear on the recommendation process in a straightforward way and it is easy to perform post-hoc credit assignment and adjust the hybrid accordingly. However, the implicit assumption in this technique is that the relative value of the different techniques is more or less uniform across the space of possible items. From the discussion above, we know that this is not always so: a collaborative recommender will be weaker for those items with a small number of raters.

3.2. Switching

A switching hybrid builds in item-level sensitivity to the hybridization strategy: the system uses some criterion to switch between recommendation techniques. The DailyLearner system uses a content/collaborative hybrid in which a content-based recommendation method is employed first. If the content-based system cannot make a

Table III: Hybridization Methods

Hybridization method	Description
Weighted	The scores (or votes) of several recommendation techniques are combined together to produce a single recommendation.
Switching	The system switches between recommendation techniques depending on the current situation.
Mixed	Recommendations from several different recommenders are presented at the same time
Feature combination	Features from different recommendation data sources are thrown together into a single recommendation algorithm.
Cascade	One recommender refines the recommendations given by another.
Feature augmentation	Output from one technique is used as an input feature to another.
Meta-level	The model learned by one recommender is used as input to another.

recommendation with sufficient confidence, then a collaborative recommendation is attempted.³ This switching hybrid does not completely avoid the ramp-up problem, since both the collaborative and the content-based systems have the “new user” problem. However, DailyLearner’s content-based technique is nearest-neighbor, which does not require a large number of examples for accurate classification.

What the collaborative technique provides in a switching hybrid is the ability to cross genres, to come up with recommendations that are not close in a semantic way to the items previous rated highly, but are still relevant. For example, in the case of DailyLearner, a user who is interested in the Microsoft anti-trust trial might also be interested in the AOL/Time Warner merger. Content matching would not be likely to recommend the merger stories, but other users with an interest in corporate power in the high-tech industry may be rating both sets of stories highly, enabling the system to make the recommendation collaboratively.

DailyLearner’s hybrid has a “fallback” character – the short-term model is always used first and the other technique only comes into play when that technique fails. Tran & Cohen (1999) proposed a more straightforward switching hybrid. In their system, the agreement between a user’s past ratings and the recommendations of each technique are used to select the technique to employ for the next recommendation.

Switching hybrids introduce additional complexity into the recommendation process since the switching criteria must be determined, and this introduces another level of parameterization. However, the benefit is that the system can be sensitive to the strengths and weaknesses of its constituent recommenders.

3.3. Mixed

Where it is practical to make large number of recommendations simultaneously, it may be possible to use a “mixed” hybrid, where recommendations from more than one technique are presented together. The PTV system (Smyth and Cotter 2000) uses this approach to assemble a recommended program of television viewing. It uses content-based techniques based on textual descriptions of TV shows and collaborative information about the preferences of other users. Recommendations from the two techniques are combined together in the final suggested program.

The mixed hybrid avoids the “new item” start-up problem: the content-based component can be relied on to recommend new shows on the basis of their descriptions even if they have not been rated by anyone. It does not get around the “new user” start-up problem, since both the content and collaborative methods need some data about user preferences to get off the ground, but if such a system is integrated into a digital television, it can track what shows are watched (and for how long) and build its profiles accordingly. Like the fallback hybrid, this technique has the desirable “niche-finding” property in that it can bring in new items that a strict focus on content would eliminate.

The PTV case is somewhat unusual because it is using recommendation to assemble a composite entity, the viewing schedule. Because many recommendations are needed to fill out such a schedule, it can afford to use suggestions from as many sources as possible. Where conflicts occur, some type of arbitration between methods is

³ Actually Billsus’ system has two content-based recommendation algorithms, one short-term and one long-term, and the fallback strategy is short-term/collaborative/long-term.

required – in PTV, content-based recommendation take precedence over collaborative responses. Other implementations of the mixed hybrid, ProfBuilder (Wasfi, 1999) and PickAFlick (Burke et al. 1997; Burke, 2000), present multiple recommendation sources side-by-side. Usually, recommendation requires ranking of items or selection of a single best recommendation, at which point some kind of combination technique must be employed.

3.4. Feature Combination

Another way to achieve the content/collaborative merger is to treat collaborative information as simply additional feature data associated with each example and use content-based techniques over this augmented data set. For example, Basu, Hirsh & Cohen (1998) report on experiments in which the inductive rule learner Ripper was applied to the task of recommending movies using both user ratings and content features, and achieved significant improvements in precision over a purely collaborative approach. However, this benefit was only achieved by hand-filtering content features. The authors found that employing all of the available content features improved recall but not precision.

The feature combination hybrid lets the system consider collaborative data without relying on it exclusively, so it reduces the sensitivity of the system to the number of users who have rated an item. Conversely, it lets the system have information about the inherent similarity of items that are otherwise opaque to a collaborative system.

3.5. Cascade

Unlike the previous hybridization methods, the cascade hybrid involves a staged process. In this technique, one recommendation technique is employed first to produce a coarse ranking of candidates and a second technique refines the recommendation from among the candidate set. The restaurant recommender EntreeC, described below, is a cascaded knowledge-based and collaborative recommender. Like Entree, it uses its knowledge of restaurants to make recommendations based on the user's stated interests. The recommendations are placed in buckets of equal preference, and the collaborative technique is employed to break ties, further ranking the suggestions in each bucket.

Cascading allows the system to avoid employing the second, lower-priority, technique on items that are already well-differentiated by the first or that are sufficiently poorly-rated that they will never be recommended. Because the cascade's second step focuses only on those items for which additional discrimination is needed, it is more efficient than, for example, a weighted hybrid that applies all of its techniques to all items. In addition, the cascade is by its nature tolerant of noise in the operation of a low-priority technique, since ratings given by the high-priority recommender can only be refined, not overturned.

3.6. Feature Augmentation

One technique is employed to produce a rating or classification of an item and that information is then incorporated into the processing of the next recommendation technique. For example, the Libra system (Mooney & Roy 1999) makes content-based recommendations of books based on data found in Amazon.com, using a naive Bayes text classifier. In the text data used by the system is included "related authors" and "related titles" information that Amazon generates using its internal collaborative systems. These features were found to make a significant contribution to the quality of recommendations.

The GroupLens research team working with Usenet news filtering also employed feature augmentation (Sarwar et al. 1998). They implemented a set of knowledge-based "filterbots" using specific criteria, such as the number of spelling errors and the size of included messages. These bots contributed ratings to the database of ratings used by the collaborative part of the system, acting as artificial users. With fairly simple agent implementations, they were able to improve email filtering.

Augmentation is attractive because it offers a way to improve the performance of a core system, like the NetPerceptions' GroupLens Recommendation Engine or a naive Bayes text classifier, without modifying it. Additional functionality is added by intermediaries who can use other techniques to augment the data itself. Note that this is different from feature combination in which raw data from different sources is combined.

While both the cascade and augmentation techniques sequence two recommenders, with the first recommender having an influence over the second, they are fundamentally quite different. In an augmentation hybrid, the features used by the second recommender include the output of the first one, such as the ratings contributed by GroupLens' filterbots. In a cascaded hybrid, the second recommender does not use any output from the first recommender in producing its rankings, but the results of the two recommenders are combined in a prioritized manner.

3.7. Meta-level

Another way that two recommendation techniques can be combined is by using the model generated by one as the input for another. This differs from feature augmentation: in an augmentation hybrid, we use a learned model to generate features for input to a second algorithm; in a meta-level hybrid, the entire model becomes the input. The first meta-level hybrid was the web filtering system Fab (Balabanovic 1997, 1998). In Fab, user-specific selection agents perform content-based filtering using Rocchio's method (Rocchio 1971) to maintain a term vector model that describes the user's area of interest. Collection agents, which garner new pages from the web, use the models from all users in their gathering operations. So, documents are first collected on the basis of their interest to the community as a whole and then distributed to particular users. In addition to the way that user models were shared, Fab was also performing a cascade of collaborative collection and content-based recommendation, although the collaborative step only created a pool of documents and its ranking information was not used by the selection component.

A meta-level hybrid that focuses exclusively on recommendation is described by Pazzani (1999) as "collaboration via content". A content-based model is built by Winnow (Littlestone & Warmuth 1994) for each user describing the features that predict restaurants the user likes. These models, essentially vectors of terms and weights, can then be compared across users to make predictions. More recently, Condliff et al. (1999) have used a two-stage Bayesian mixed-effects scheme: a content-based naive Bayes classifier is built for each user and then the parameters of the classifiers are linked across different users using regression. LaboUr (Schwab, et al. 2001) uses instance-based learning to create content-based user profiles which are then compared in a collaborative manner.

The benefit of the meta-level method, especially for the content/collaborative hybrid is that the learned model is a compressed representation of a user's interest, and a collaborative mechanism that follows can operate on this information-dense representation more easily than on raw rating data.

3.8. Summary

Hybridization can alleviate some of the problems associated with collaborative filtering and other recommendation techniques. Content/collaborative hybrids, regardless of type, will always demonstrate the ramp-up problem since both techniques need a database of ratings. Still, such hybrids are popular, because in many situations such ratings already exist or can be inferred from data. Meta techniques avoid the problem of sparsity by compressing ratings over many examples into a model, which can be more easily compared across users. Knowledge-based and utility-based techniques seem to be good candidates for hybridization since they are not subject to ramp-up problems.

Table IV summarizes some of the most prominent research in hybrid recommender systems. For the sake of simplicity, the table combines knowledge-based and utility-based techniques (since utility-based recommendation is a special case of knowledge-based).⁴

There are four hybridization techniques that are order-insensitive: Weighted, Mixed, Switching and Feature Combination. With these hybrids, it does not make sense to talk about the order in which the techniques are applied: a CN/CF mixed system would be no different from a CF/CN one. The redundant combinations are marked in gray.

The cascade, augmentation and meta-level hybrids are inherently ordered. For example, a feature augmentation hybrid that used a content-based recommender to contribute features to be used by a second collaborative process, would be quite different from one that used collaboration first. To see the difference, consider the example of news filtering: the former case, content-based/collaborative, would correspond to a learning content-based version of the GroupLens "filterbot" idea. The latter arrangement, collaborative/content-based, could be implemented as a collaborative system that assigns users to a clique or cluster of similar users and then uses the clique ids as input to a content-based system, using these identifiers as well as terms from the news articles to produce the final recommendation. We would expect these systems to have quite different characteristics. With cascade, feature augmentation and meta-level hybrids therefore all permutations must be considered and these columns do not contain any redundancies.

There are 60 non-redundant spaces in the table, but some combinations are not possible. Since a knowledge-based technique may take into account any kind of data, feature combination does not really represent a possible hybrid.

⁴ We ignore hybrids that combine techniques of the same type, although some do exist. PickAFlick (Burke et al. 1997), for example, is a knowledge-based/knowledge-based mixed hybrid combining two different knowledge-based strategies.

Conversely, the demographic technique is similar to the collaborative in its approach (comparing users against each other), just using different features (demographic data vs. ratings) to do so. Therefore, it does not make sense to distinguish a content-based/demographic (CN/DM) meta-level hybrid from a content-based/collaborative (CN/CF) one. The illogical hybrids are marked in black. The white areas of the table enumerate 53 different possible hybrid recommender systems. Of the possible hybrids, only 14 seem to have been explored, leaving significant room for further research.

This chart suggests some interesting types of recommenders that do not yet exist. Although collaborative filtering is the most fully explored technique, a number of its hybrids remain unexplored.



- Content-based/collaborative feature augmentation hybrid. This possibility was described earlier: a content-based “filterbot”.
- Collaborative/content-based meta-level hybrid, in which collaborative information is used to generate a representation of overall user ratings for an item and this representation is then used to compare across items.
- Collaborative/demographic augmentation hybrid in which a collaborative technique is used to place the user in a niche of like-minded users, and this information is used as a feature in a demographic rater.
- In addition, four cascade recommenders involving collaborative recommendation appear untried.

Other techniques show even fewer examples. Demographic techniques are poorly represented because this kind of data is more difficult to obtain than user ratings. Only 4 of the possible 25 such hybrids appear to have been attempted. Knowledge- and utility-based techniques are also relatively under-explored with 4 of the possible 26 of these combinations researched. Together these techniques account for 36 of the 39 possible hybrid recommenders not yet explored. One reason for this focus on collaborative and content-based techniques is the availability of ratings databases, such as the popular EachMovie database, which has approximately 45,000 ratings for 250 users. When combined with public data on movies, this database has enabled researchers to explore content-based and

Table IV: Possible and Actual (or Proposed) Recommendation Hybrids

	Weighted	Mixed	Switching	Feature Combination	Cascade	Feature Aug.	Meta-level
CF/CN	P-Tango	PTV, ProfBuilder	DailyLearner	(Basu, Hirsh & Cohen 1998)	Fab	Libra	
CF/DM	(Pazzani 1999)						
CF/KB	(Towle & Quinn 2000)		(Tran & Cohen, 2000)				
CN/CF							Fab, (Condliff, et al. 1999), LaboUr
CN/DM	(Pazzani 1999)			(Condliff, et al. 1999)			
CN/KB							
DM/CF							
DM/CN							
DM/KB							
KB/CF					EntreeC	GroupLens (1999)	
KB/CN							
KB/DM							

(CF = collaborative, CN = content-based, DM = demographic, KB = knowledge-based / utility-based)

 Redundant
 Not possible

collaborative techniques quite thoroughly.

While the space remains to be fully explored, research has provided some insight into the question of which hybrid to employ in particular situations. The hybridization strategy must be a function of the characteristics of the recommenders being combined. With demographic, content and collaborative recommenders, this is largely a function of the quality and quantity of data available for learning. With knowledge-based recommenders, it is a function of the available knowledge base. We can distinguish two cases: the uniform case, in which one recommender has better accuracy than another over the whole space of recommendation, and the non-uniform case, in which the two recommenders have different strengths in different parts of the space. If the recommenders are uniformly unequal, it may make sense to employ a hybrid in which the inaccuracies of the weaker recommender can be contained: for example, a cascade scheme with the stronger recommender given higher priority, an augmentation hybrid in which the weaker recommender acts as a “bot” contributing a small amount of information, or a meta-level combination in which the stronger technique produces a dense representation that strengthens the performance of the weaker one. In the non-uniform case, the system will need to be able to employ both recommenders at different times. A switching hybrid is a natural choice here, but it requires that the system be able to detect when one recommender should be preferred. Feature combination and mixed hybrids can be used to allow output from both recommenders without having to implement a switching criterion. More research is needed to establish the tradeoffs between these hybridization options.

4. A Knowledge-Based Restaurant Recommender System

As the overview shows, there are a number of areas where the space of hybrid recommendation is not fully explored. In particular, there are few examples that incorporate knowledge-based recommendation. Knowledge-based recommendation is at the heart of a research program known as “Find-Me Systems” (Burke et al. 1997; Burke 1999a; Burke, 2000). The restaurant recommender Entree is one example of such a system. This section provides a brief overview of Entree, and then introduces EntreeC, a hybrid recommender system that adds collaborative filtering to Entree, creating a knowledge-based/collaborative cascade hybrid.

4.1. Entree

Entree is a restaurant recommendation system that uses case-based reasoning (Kolodner 1993) techniques to select and rank restaurants. It was implemented to serve as a guide to attendees of the 1996 Democratic National Convention in Chicago and has been operating as a web utility since that time.⁵

A user interacts with the system by submitting an entry point, either a known restaurant or a set of criteria, and is shown similar restaurants. The user then interacts with the system in a dialog, critiquing the system’s suggestions and interactively refining the search until an acceptable option is achieved. Consider a user who starts browsing by entering a query in the form of a known restaurant, Wolfgang Puck’s “Chinois on Main” in Los Angeles, as shown in Figure 1. (The user may also make a database query based on desired restaurant features.) As shown in Figure 2, the system finds a similar Chicago restaurant that combines Asian and French influences, “Yoshi’s Cafe,” as well as other similar restaurants that are ranked by their similarity. Note that the connection between “Pacific New Wave” cuisine and its Asian and French culinary components is part of the system’s knowledge base of cuisines. The user, however, is interested in a cheaper meal and selects the “Less \$\$” button. The result shown in Figure 3 is a creative Asian restaurant in a cheaper price bracket: “Lulu’s.” However, the French influence is lost — one consequence of the move to a lower price bracket. The user can continue browsing and critiquing until an acceptable restaurant has been located.

A key consideration in the design of the system was to support a natural interactive retrieval process. After the first page, no action requires more than a single click, an important consideration in efficient use of the web medium. The system presents one main result and a small number of neighbors rather than an overwhelming list, and the user can explore the space of restaurants, discovering for example, the tradeoffs between price and quality for restaurants serving a given cuisine.

⁵ <URL: <http://infolab.ils.nwu.edu/entree/>>

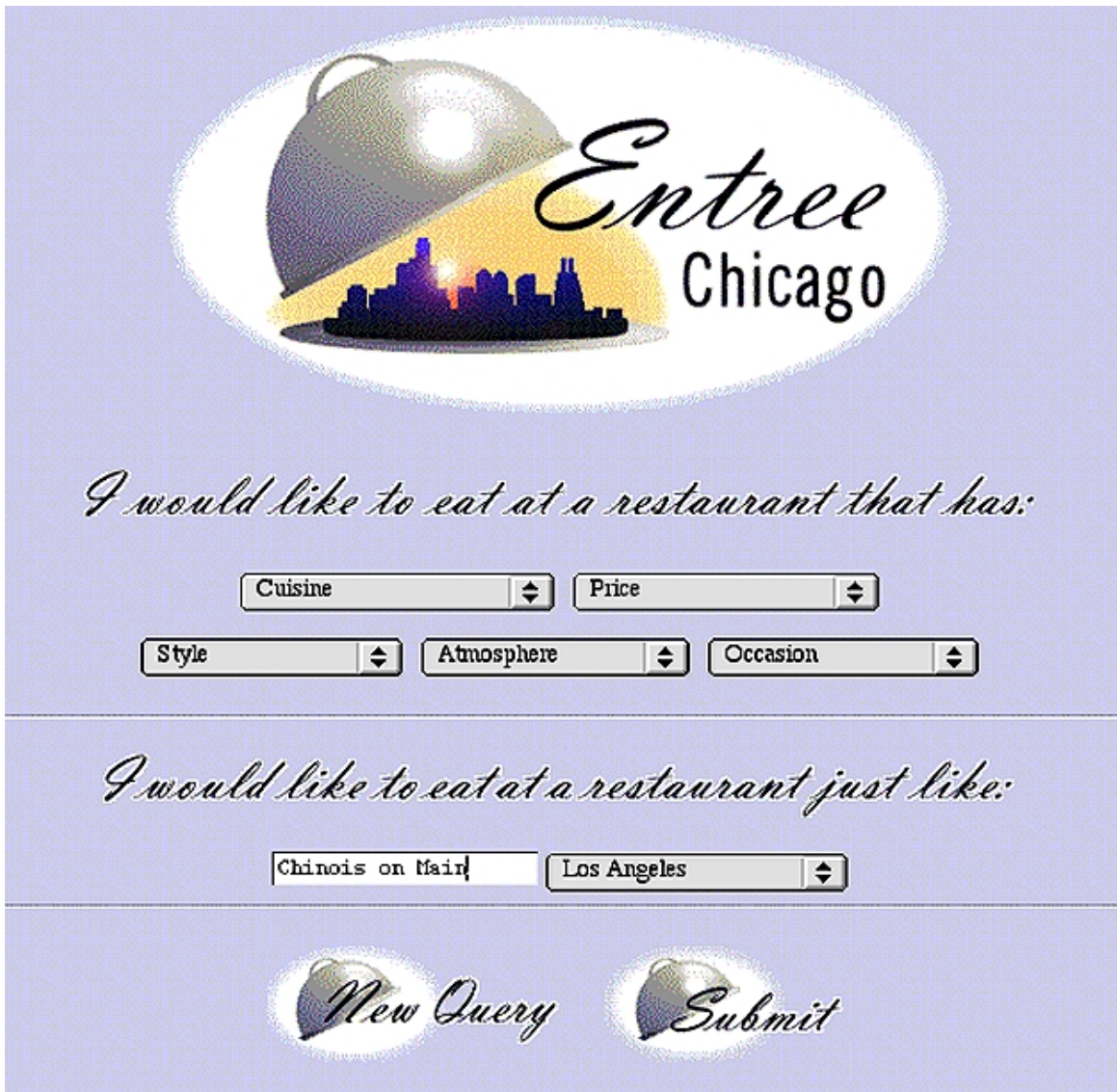


Figure 1. The Entree Restaurant Recommender: Initial screen

Entree's recommendation technique is one of knowledge-based similarity retrieval. There are two fundamental retrieval modes: similarity-finding and critique-based navigation. In the similarity case, the user has selected a given item from the catalog (called the *source*) and requested other items similar to it. To perform this retrieval, a set of candidate entities is retrieved from the database, sorted based on similarity to the source and the top few candidates returned to the user. Navigation is essentially the same except that the candidate set is filtered prior to sorting to leave only those candidates that satisfy the user's critique. For example, if a user responds to item X with the tweak "Nicer," the system determines the "niceness" value of X and rejects all candidates except those whose value is greater. Entree does not retain a user profile as such. It is a stateless application – its response completely determined by the example to which the user is responding and the specific critique given.



Entree Results

The Los Angeles restaurant you chose is:

Chinois On Main	
2709 Main St. (bet. Rose Ave. & Ocean Park Blvd.), Santa Monica, 310-392-9025	
Pacific New Wave	\$30-\$50
Extraordinary Decor, Extraordinary Service, Near-perfect Food, Hip Place To Be, On the Beach, Great for People Watching, Parties and Occasions, Weekend Brunch, Weekend Lunch, Fabulous Wine Lists	

We recommend:

Yoshi's Cafe	
3257 N. Halsted St. (Belmont Ave.), Chicago, 312-248-6160	
Asian, Japanese, French (New)	\$30-\$50
Extraordinary Decor, Extraordinary Service, Near-perfect Food, Need To Dress, Prix Fixe Menus, Quiet for Conversation, Very Busy - Reservations a Must, Romantic, Good Out of Town Business, Fabulous Wine Lists, Game, Parking/Valet	



For other suggestions, select:

[Yoshi's Cafe](#)

[Penny's Noodle Shop](#)

[Emilio's Tapas Bar & Restaurant](#)

[Emilio's Granada](#)

[302 West](#)

[Arun's](#)

[Nick's Fishmarket](#)

[Lulu's](#)

[Trio](#)

[Bossa Nova](#)

Figure 2. Similarity-based Recommendation

A case-based reasoning (CBR) system is a problem solver that uses the recall of examples as the fundamental problem-solving process (Kolodner, 1990). A case-based recommender system is one that treats the objects to be recommended as cases, and employs CBR techniques to locate them. A case-based reasoning system contains a number of different “knowledge containers” (Richter, 1995): the *case base*, the *vocabulary* in which cases are described, the *similarity measure* used to compare cases, and, if necessary, the knowledge needed to *transform* recalled solutions. In building a case-based system, the developer can choose where in the system different types of knowledge can reside. A low-level vocabulary for cases may push more complexity and hence more knowledge into the similarity measure, for example. The restaurants in Entree use a simple representational vocabulary derived directly from the features present in the textual restaurant guide from which they were derived, and as a result have somewhat complex similarity metrics.

The similarity relation between restaurants is decomposed into a set of independent attributes, such as the “niceness” of a restaurant, that correspond to users’ high-level perception or interest in the object. For each such attribute, a *local similarity metric* is defined, which measures how similar two items are with respect to that attribute. Two restaurants with the same price would get the maximum similarity rating on the metric of price, but may differ




For a cheaper restaurant than:

Yoshi's Cafe	
3257 N. Halsted St. (Belmont Ave.), Chicago, 312-248-6160	
Asian, Japanese, French (New)	\$30-\$50

We recommend:

Lulu's (map)	
626 Davis St. (bet. Chicago & Orrington Aves.), Evanston, 708-869-4343	
Japanese, Asian	below \$15
Good Decor, Excellent Service, Excellent Food, Creative, No Reservations, Weekend Brunch, Wheelchair Access, Long Drive	



For other suggestions, select:

Lulu's	Penny's Noodle Shop	Sanko
Noodle Noodle	Benihana of Tokyo	Honda
New Japan	Hatsuhana	Daruma
Kampai	Akai Hana	

Figure 3. Critique-based Navigation (“Less \$\$”)

greatly on another metric, such as quality or type of cuisine. Each local metric has a small range of integer values as its output.

Once the similarity relations are defined, they are ordered to create a *retrieval strategy*. For example, in the Entree restaurant recommender system, the attributes were cuisine, price, quality, and atmosphere applied in rank order. The metrics are combined in a cascaded way: each subsequent metric used only to break ties between restaurants ranked equally by higher level metrics. One benefit of combining the local metrics in this way is that the global metric is much simpler to design and its behavior easier to predict, as compared to the weighted combinations often found in CBR systems.

The actual knowledge content of Entree’s metric is fairly shallow. It knows that all other things being equal, a more expensive restaurant is worse, and a more highly-rated restaurant is better. Its most complex metric is the one that compares restaurants on the basis of cuisine. This metric is based on a semantic network containing approximately 150 different cuisines. Similarity between cuisines is represented as the inverse of distance in the network.

Entree differs from other recommender systems in several ways. Most strikingly, it uses *semantic ratings*, evaluations that tell the system not just the user’s preference – thumbs up or thumbs down – but also the reason behind the rating: too expensive, not fancy enough, etc. The semantic ratings give Entree a level of interactivity not found in other recommender systems, and allow it to tailor its suggestions to the user’s particular need of the moment, rather than adapting to overall preferences over time. The CoFIND system (Dron et al. 1999) also used

semantic ratings, allowing users to create dimensions along which to rate web pages as “good for beginners”, “comprehensive”, etc. In CoFIND, however, these assessments were not used to navigate between options.

Relevance feedback (Salton & McGill, 1983) is a well-known technique in information retrieval in which a user’s query is refined by taking into account responses to returned items. Entree’s critique-based navigation differs from relevance feedback approaches in both explicitness and flexibility. In relevance feedback approaches, the user selects some retrieved documents as being more relevant than others, and the system determines how a query is refined. In FindMe systems, critiques supply concrete domain-specific feedback that adjusts the search in a particular direction.

One problem that became apparent when Entree went into operation was the issue of inadequate discrimination (Burke 1999b). At times, the retrieval process returns a large set of items all of which are equivalent as far as the system’s similarity metric can determine. Since the system presents only a small number (to avoid overwhelming the user), it essentially picks randomly among this set, and some equally good items are never shown. To achieve finer levels of discrimination requires more knowledge: either enhanced case data (difficult or sometimes impossible to obtain) or the engineering of a more fine-grained similarity metric. Part of the appeal of building a hybrid recommender with Entree was that it offered the possibility of improving the system’s discrimination without requiring the need for additional knowledge engineering.

4.3. EntreeC: A Knowledge-Based/Collaborative Cascade Hybrid

A review of Table IV shows seven possible hybrids between knowledge-based (KB) and collaborative (CF) recommenders. The switching and mixed hybrids are appropriate in a non-uniform case, but we did not find that Entree’s recommendations were weaker in any one part of the recommendation space. In addition, we wanted to maintain the conversational interaction that is the hallmark of FindMe systems. That meant that recommendation had to be seen as a direct response to the user’s last critique rather than something more holistic. Thus, neither a KB/CF meta-level hybrid nor a KB/CF feature augmentation would have been appropriate, since in both of these configurations, it is the collaborative part this is actually recommending. CF/KB feature augmentation and CF/KB meta-level hybrids would have been possible, but in the either case, the knowledge-based part of the system would have had to make inferences from collaboratively-generated features. This would have entailed additional knowledge engineering – exactly what we sought to avoid. That leaves the weighted/cascade category of hybrids. Since Entree’s similarity assessment technique already used a cascade, we found it most elegant to simply add collaborative recommendation as a final cascaded step.

To see how such a system would differ from the existing Entree, consider the following example: Alice connects to EntreeC, a version of Entree that includes a collaborative filtering component. She starts browsing for Chicago restaurants by entering the name of her favorite restaurant at home, *Greens Restaurant* in San Francisco. Greens is characterized as serving “Californian” and “Vegetarian” cuisine. The top recommendation is *302 West*, which serves “Californian” and “Seafood.” It turns out that Alice is, in fact, a vegetarian, so she critiques the system’s cuisine choice and moves back towards vegetarian recommendations.

After the system has built up a bigger user base, another new user Bob approaches the system with the same starting point: *Greens*. Since the recommendation given to Alice was under-discriminated, her feedback and that of other users allow the system to more fully discriminate Bob’s recommendation, and return *Jane’s*, a vegetarian restaurant, now preferring it over *302 West*.

This thought experiment suggests that a cascade using both knowledge-based and collaborative-filtering techniques may produce a recommender system with some of the best characteristics of both. Initial suggestions are good, since there is a knowledge base to rely on. As the system’s database of ratings increases, it can move beyond the knowledge base to characterize users more precisely. Because the knowledge base is always present, users are not trapped by their past behavior. If Alice decides to stop being a vegetarian, she will be able to get recommendations for steakhouses by entering one as a starting point.

5. Experiments

To evaluate the potential benefit to be achieved with EntreeC, we performed a series of experiments using historical data from the use of Entree. Entree has been in continuous operation as a public web utility since July 1996. The experiments described below use logs through June 1999. For performance reasons, Entree was implemented without internal logging, and it does not use cookies or other mechanisms to identify a user returning to the site. The data for our experiments therefore comes from the logs of the web server invoking Entree, rather than from the system itself. All of the data discussed below is of the short-term variety, representing a single search attempt by a user. To create

these sessions, the web server's log data was partitioned into sessions, identified by IP address and terminating after 10 minutes of inactivity. There are approximately 50,000 sessions in the 3 years of usage data.⁶

Because only short-term data is available, these experiments may show something of a lower bound on the efficacy of collaborative filtering: a site would normally be expected to gather user ratings over multiple sessions of interaction and tie them directly to a user identifier, enabling the construction of a long-term user profile. On the other hand, ratings gathered over a longer period of time would reflect a diversity of search goals, a diversity presumably not present in a single search session. Further research with EntreeC will explore this long-term/short-term profile trade-off. (See Section 6 below.)

Each session consists of a list of user actions and associated restaurants. As shown in the screen shot, a number of restaurants are retrieved as likely candidates, but one is highlighted as the most similar item. It is to this item that the user can respond with a critique. There are eight possible navigation actions a user can take: "Less \$\$," "Nicer," "More Creative," "More Traditional," "Quieter," "Livelier," "Change Cuisine,"⁷ and "Browse" (the choice to move to a different restaurant in the return list.) A user can begin the session with a known restaurant as a starting point or with a query that describes the type of restaurant sought, but again, due to constraints on the implementation of the original system, these queries were also not logged. So, for each restaurant, we can associate one of 10 actions: Entry point, Exit point, or one of the eight critiques.

The sessions range in length from one to 20 interactions, but typically contain less than ten. On occasion, the same restaurant is rated more than once. For example, a user might see a recommendation, browse to the other restaurants in the list, return to the original suggestion, and then perform a critique. We discard all but the most recent rating for each restaurant.

5.1. Navigation Actions as Implicit Ratings

The research reported here attempts to estimate how much improvement might be expected from adding collaborative filtering to Entree, and determine what collaborative technique would produce the best performance. The central issues are the generation of ratings and the computation of inter-user similarity. The actions that users take in the Entree interface are intended primarily for navigation among options, not primarily to record evaluations of objects, but they can be considered implicit ratings.

With critiques, we can be fairly certain of the type of rating that should be inferred, both in terms of its negative valence – "give me something else" – and in its meaning – "this one is too expensive." Table V shows Nichols (1997) proposed scale for interpreting the relative importance of different types of actions interpreted as ratings. The idea behind this table is to rate the significance that should be afforded to different types of implicit feedback. "Purchase" is the most significant action: if the user bought something, we should be confident that she liked it. On the other hand, if the item was only contained on a page that the user saw ("Glimpse"), we can have much less confidence. A critique is an assessment, which comes second only to "Purchase" in the list, since the user must consider the merits of a suggestion and respond to it. For example, if user A sees "Yoshi's Cafe" and clicks "Less \$\$," this can be confidently recorded as a negative rating.

However, similar certainty is difficult to achieve in terms of positive ratings. If a user enters a restaurant as a starting point, we consider this a "referring" action: the user makes references to a known restaurant to find something similar. So, the Entry action is fairly certain to indicate a positive rating. If the user stops browsing upon encountering some restaurant, we assume that the desired answer has been found. This "Stops looking" datum is a weaker data point (perhaps falling between "Examine" and "Mark" on Nichols' scale), since it is also possible that the user has failed to find a restaurant and has given up.

5.2. Collaborative Techniques

If we accept navigation actions as implicit ratings, they can be easily turned into numeric ratings of the type used by collaborative algorithms. Many collaborative systems work with simple positive/negative ratings: in our case, Entry and Exit would be positive, and all others negative. If we wish the numeric ratings to reflect something of the semantics of the user's original action, we can have a more graduated version conversion scale:

⁶ The Entree data set is available at the UCI KDD Archive <URL: <http://kdd.ics.uci.edu/>>.

⁷ In the case of the user choosing an alternative cuisine, there is a separate step in which the alternate cuisine is chosen. The retrieval and filtering operations are the same as for the other critiques, the log does not record the user's cuisine choice.

Table V. Implicit Ratings in Order of Strength after Nichols (1997). (Addition in italics.)

	Action
1	Purchase
2	Assess
3	Repeated Use
4	Save / Print
5	Delete
6	Refer
7	Reply
8	Mark
9	<i>Terminate Search</i>
10	Examine / Read
11	Consider
12	Glimpse
13	Associate
14	Query

- Entry point: The user typed this restaurant in as a starting point so we will assume it is one they liked. Rating = 1.0
- Exit point: The user stopped here. Maybe they found what they sought, but possibly they gave up. Rating = 0.8
- Browse: There is no direct critique of the restaurant, but the user is moving away from it. Rating = -0.5.
- Critiquing: The user is giving direct feedback that there is something undesirable about the restaurant. Rating = -1.0

Assume that we have the log of an interactive session with user A, S_A , consisting of restaurant/navigation action pairs: $\langle r, a \rangle$. This session can be converted to a vector of ratings by substituting the appropriate numeric values for the ratings to yield a rating vector R_A . Once all sessions have been similarly transformed, we can compare two users using standard collaborative filtering algorithms such as Pearson's correlation coefficient. The correlation between two users A and B would be given by

$$r(A, B) = \frac{n(\sum R_A R_B) - (\sum R_A)(\sum R_B)}{\sqrt{(n\sum R_A^2 - (\sum R_A)^2)(n\sum R_B^2 - (\sum R_B)^2)}}$$

When an restaurant is rated in one session, but not in the other, a default score of 0 is given, a technique that has been shown to increase accuracy over discarding such ratings (Breese *et al.*, 1998).

Although a simple numeric conversion of the navigation actions into ratings satisfies the input criteria of collaborative filtering algorithms, it is ultimately unsatisfying from a user modeling point of view. The conversion loses entirely the reason for the user's preference, which has been explicitly given to the system. Suppose Alice looks at "Chez Frou-Frou" and selects the "Less \$\$" button and Bob looks at the same restaurant and clicks on "Nicer". The implicit rating approach would count these ratings as evidence that Alice and Bob are similar users, something that the semantics of the ratings would suggest is unlikely.

To avoid losing the semantic details of ratings, we can perform collaborative filtering by looking only for users with exactly the same ratings: treating ratings on different dimensions as incommensurable. We would only match Alice against others who also thought that "Chez Frou-Frou" was too expensive. The problem then becomes a multi-dimensional vector-matching problem of the type frequently encountered in information retrieval contexts (Salton & McGill 1983). We convert the session to a vector V where each restaurant / rating combination is a separate dimension. Alice's session will show a 1 in the position corresponding to the "Chez Frou-Frou"/"Less \$\$" combination, and Bob's session would have a 0 there and a 1 in the "Chez Frou-Frou"/"Nicer" dimension. With each user represented as a multi-dimensional vector, we can use the cosine measure of similarity.

Table VI. Similarity Matrix for Entree Ratings

Br.	Ch.	Ni.	Tr.	Cr.	Li.	Qu.	Cu.	En.	Ex	
1	0	0	0	0	0	0	0	0	0	Browse
	1	-1	-0.5	-0.5	-0.5	-0.5	0	0	0	Cheaper
		1	0.5	0.5	-0.5	0.5	0	0	0	Nicer
			1	-1	-0.5	0.5	0	0	0	Trad.
				1	0.5	-0.5	0	0	0	Creat.
					1	-1	0	0	0	Lively
						1	0	0	0	Quiet
							1	0	0	Cuisine
								1	1	Entry
									1	Exit

$$\cos(A, B) = \frac{\sum V_A V_B}{\sum V_A^2 \cdot \sum V_B^2}$$

However, with several thousand restaurants and 10 possible ratings, we have a high-dimensional vector space and a highly sparse comparison problem: we will only compare users whose reactions are largely identical, meaning that predictions will be based on a much smaller number of users than in the single-scale collaborative approach.

A third technique takes into account the semantics of the ratings themselves: we establish a similarity metric between ratings based on their characteristics. In the example above, we should probably rate Alice and Bob as dissimilar even though they both disliked the same restaurant – they did so for essentially opposite reasons. This is a “heuristic similarity” approach, a familiar idea in case-based reasoning. It does not establish a single numeric scale to which all actions are converted, but rather looks at the similarity of users on a rating by rating basis. This metric takes the qualitative differences between ratings into account, but it allows more kinds of inter-user comparison than the sparse metric. A similarity value is assigned to each possible pair of ratings, using an adjacency table generated by considering the semantics of each response type, and a few common-sense considerations:

- A rating is maximally similar to itself.
- “Browse” is not similar to any other rating.
- Some ratings have natural opposites: “Livelier” / “Quieter”, “Traditional” / “Creative.”

The full comparison table is shown in Table VI. Let the function from action pairs to distances described by this table be represented by the function $d(a_1, a_2)$. Let R be the set of restaurants that appear in both S_A and S_B . Heuristic similarity h between users is determined by averaging their distances for all restaurants rated in common.

$$h(A, B) = \frac{\sum_{r \in R} d(a, b)}{|R|} \text{ where } \langle r, a \rangle \in S_A \text{ and } \langle r, b \rangle \in S_B.$$

Experimentation with this heuristic indicates that it is relatively insensitive to the magnitude of the values in the table, but highly sensitive to their sign: it is important that “Livelier” be represented as opposite from “Quieter”.

To see the difference between the techniques, consider the following example shown in Figure 4. Alice has been looking for a restaurant using EntreeC and has accumulated the profile shown. The system has in it three other profiles: Bob, Carol, and Dan. Using correlation, computation would proceed as shown in Table VII. The profiles are transformed into numeric vectors, which are then correlated as shown in Table VIII. A look at the original data shows that Bob and Alice probably have quite different tastes in restaurants: Bob cannot find any restaurant that is nice enough for him, while Alice is often looking for a bargain. Yet Bob is the most similar user by this measure.

Using the sparse vector technique requires that we create binary vectors that are large enough to incorporate every combination of rating and restaurant. Table IX shows condensed versions of these vectors, omitting the all-zero columns. When these vectors are compared, Bob is no longer such a good match, since his actions do not match up well with Alice’s. As Table X shows, Carol is the one who comes closest, and she is certainly a better candidate than Bob. However, inspection shows certain incompatibilities that this technique does not recognize. For example, both La Italia and Chez Nouvelle are critiqued by Alice but liked by Carol.

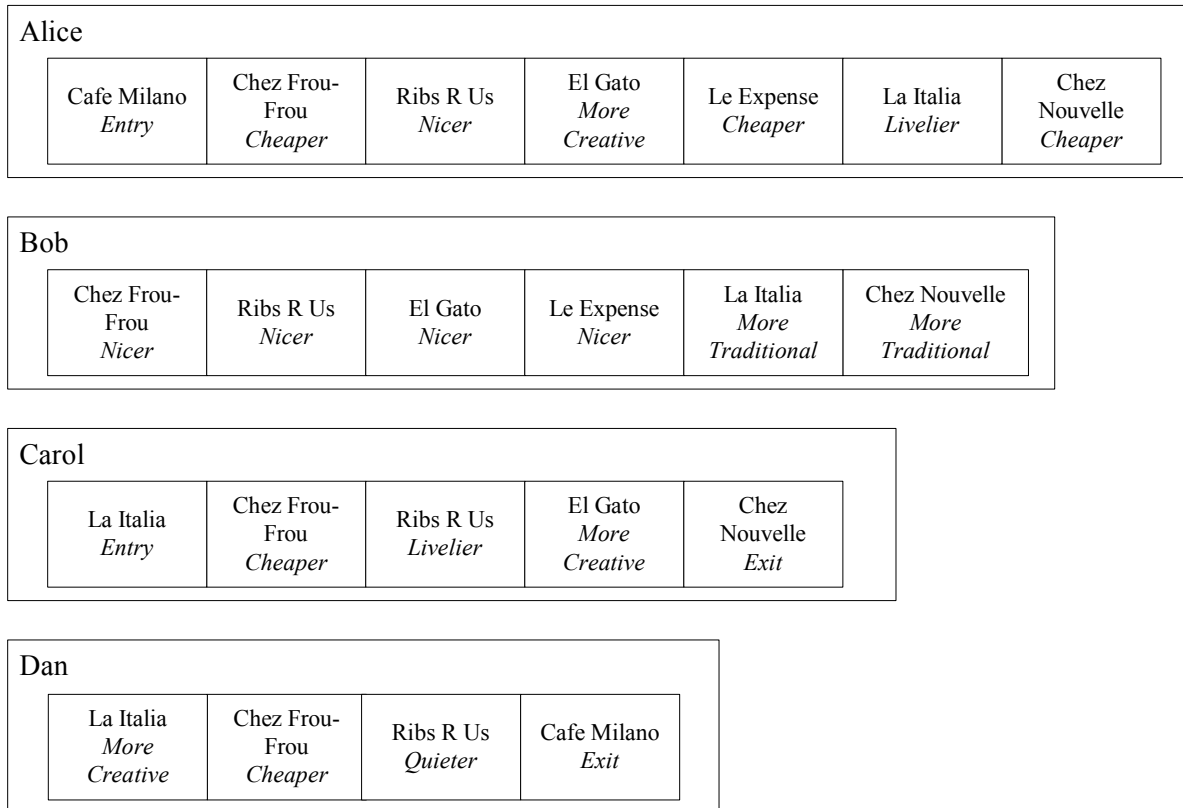


Figure 4. Sample Entree Sessions

Table VII: Profile Representation 1 – Rating Vector

	Cafe Milano	Chez Frou-Frou	Chez Nouvelle	El Gato	La Italia	Le Expense	Ribs R Us
Alice	1	-1	-1	-1	-1	-1	-1
Bob	0	-1	-1	-1	-1	-1	-1
Carol	0	-1	0.8	-1	1	0	-1
Dan	0.8	-1	0	0	-1	0	-1

Table VIII: Similarity Computation 1 – Correlation

User	Correlation
Bob	1.00
Carol	0.08
Dan	0.70

Table IX: Profile Representation 2 – Sparse Vector

	Cafe Milano		Chez Frou-Frou		Chez Nouvelle			El Gato		La Italia				Le Expense		Ribs R Us		
	Entry	Exit	Cheaper	Nicer	Cheaper	More Trad.	Exit	More Creative	Nicer	Liveller	More Trad.	Entry	More Creative	Nicer	Cheaper	Nicer	Liveller	Quieter
Alice	1	0	1	0	1	0	0	1	0	1	0	0	0	0	1	1	0	0
Bob	0	0	0	1	0	1	0	0	1	0	1	0	0	1	0	1	0	0
Carol	0	0	1	0	0	0	1	1	0	0	0	1	0	0	0	0	1	0
Dan	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1

Table X: Similarity Computation 2 – Cosine

User	Cosine
Bob	0.024
Carol	0.057
Dan	0.036

Table XI: Local and Global Heuristic Distances

	Cafe Milano	Chez Frou-Frou	Chez Nouvelle	El Gato	La Italia	Le Expense	Ribs R Us	Average distance
Bob	0	-1	-0.5	0.5	-0.5	-1	1	-0.21
Carol	0	-1	0	1	0	0	-0.5	-0.07
Dan	1	1	0	0	0.5	0	0.5	0.43

Table XII: Comparison of the Three Collaborative Techniques

Collaborative Technique	Profile conversion	Distance metric
Correlation	Actions converted to numeric preference values	Correlation
Sparse	Restaurant/action pairs converted to dimensions in a multi-dimensional vector space	Vector cosine
Heuristic	None	Average heuristic similarity

Using the third technique, the ratings are not transformed but rather compared individually using the distance measure described above. The local distances between Alice and the other users and the global averages are shown in Table XI. Here we see that the heuristic technique brings out the similarity between Dan and Alice. Dan does not have many identical ratings, but his ratings have similar semantics and suggest he may really be the most similar user.

This simple example illustrates how differences in the interpretation of the navigation actions result in different assessments of similarity between users. The three techniques, which were the subject of the experiments described below, are summarized in Table XII.

5.3. Methodology

Our evaluation methodology needed to be sensitive to the goal of the combined hybrid. With a cascade, we need the secondary, collaborative, recommender to refine the candidates left under-discriminated by the knowledge-based system. The task therefore becomes one of selecting the best candidate out of a set rather than surveying the entire universe of possible recommendations.

To evaluate the different collaborative filtering approaches, the Entree session data was partitioned into equal-sized training and test sets of approximately 25,000 sessions. The training/test split was performed five times for cross-validation. The training set was used as the profile database from which collaborative recommendations were made. From the test set, highly active users were extracted, those with at least 15 restaurants rated, about 200 users in total. (Later evaluations looked at less active users.) Again, because of the way in which actions were recorded, only short-term sessions were accessible from this data, and in order to have sufficient profile data, we could only take

```

Let S = a session: {s0, ..., sn} where each si consists of a
    pair <restaurant, action>
r = a pair with a positive action (Entry or Exit) from S
T = test data for the session, initially { }
P(S, t), a prediction function that predicts the rating of test
    item t, given ratings from profile data in S.
m = maximum predicted rating so far, initially -1
M = restaurant with maximum predicted rating, initially null

move r from S to T
move 6 random s from S to T
for each t in T
    p = P(S, t)
    if p > m
        m = p
        M = t
if M = r then
    prediction correct
else
    prediction incorrect

```

Figure 5. The Evaluation Algorithm

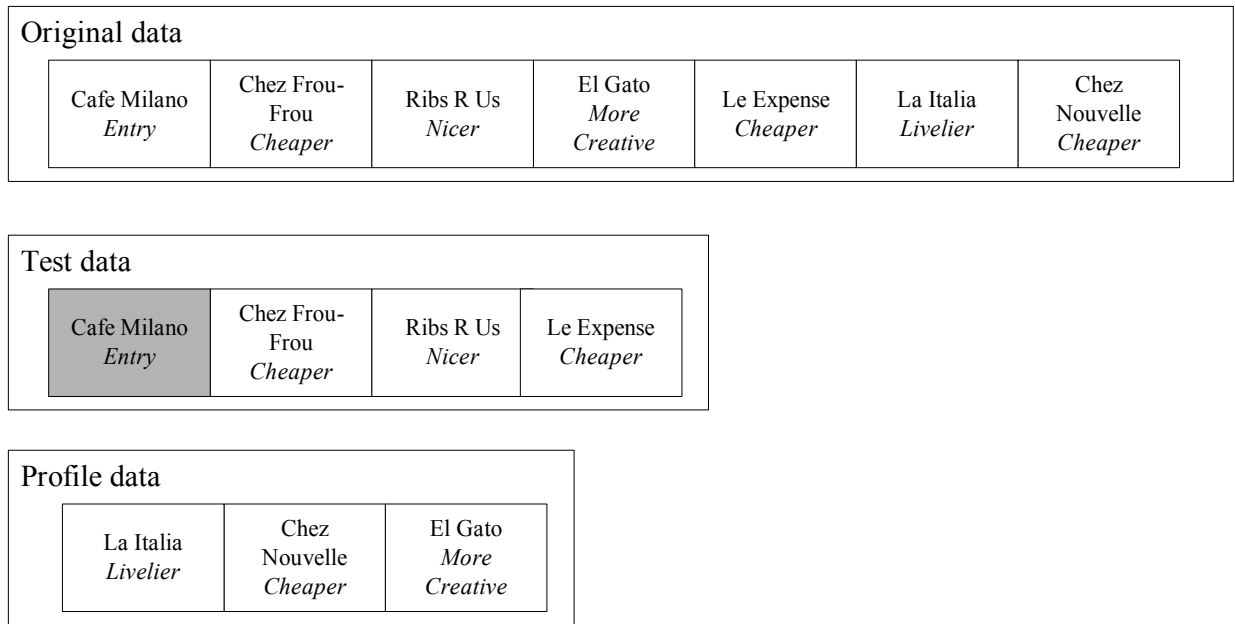


Figure 6. Evaluation Example: Test and Profile Data

advantage of those few users who examined 15 or more restaurants in a single sitting. A commercial system that used cookies to identify returning users would accumulate 15 ratings for a user very quickly.

The goal of collaborative filtering for Entree was not strictly to predict ratings, but rather to improve the quality of the recommendations made by the knowledge-based component. The evaluation technique reflects this application. See the outline of the algorithm in Figure 5. For each session S , the system first isolates a positively rated restaurant r — an item the user found satisfactory. The goal is to bring this restaurant to the user’s attention as soon as possible, so the evaluation task for the CF system is to pick this positively-rated item from a group of negatively-rated ones. To simulate this situation, the system randomly selects 6 items from S with a negative rating, and groups them with r so that there are seven restaurants in the test data T . Eight items then become profile data for a particular user, excess ratings being discarded at random. (Five such profile/test splits are generated for each session for a second level of cross-validation.) Using the profile, a prediction is made for the rating of each test item $t \in T$, and the one with the highest predicted score is selected as the recommendation. To look at each algorithm’s learning performance, we can vary the amount of profile data that is given to it, and look at differences in predictive performance.

Figure 6 illustrates a simplified version of this process for the evaluation of a single recommender on the seven items of Alice’s data. Four ratings are put into the test data set including her lone positive rating. The remaining three ratings become the profile set. As indicated in Figure 7, the recommender is then given the profile data and asked to rate each of the items in the test set. The restaurant that the recommender with highest predicted rating (in this example, “Chez Frou-Frou”) becomes the “recommendation” from the test set. This does not match Alice’s known favorite “Café Milano” and so this recommendation interaction would be counted as a failure.

For the correlation technique, predictions of the rating of a test item t are made by selecting all users who have rated t , selecting those who meet a minimum threshold of correlation with the test user based on the training data, and averaging their ratings of t . The same experimental scheme was applied also to the sparse metric, using cosine similarity. For the heuristic metric, the single nearest neighbor was used for prediction. As a baseline, we also used a predictor based on the average rating of a restaurant across all users.

For our first experiment, we examined the effect of different conversions from navigation actions to implicit ratings. We tested four recommenders: the Correlation algorithm described above with scalar-valued ratings and also the same algorithm with binary (like/dislike) ratings; and for comparison, a recommender using just the overall average rating for each restaurant also implemented with scalar and binary ratings. Figure 8 shows the accuracy of each technique (percentage of correct identifications of the user’s preferred item) as the amount of training data is increased from 4 ratings to 8. We see a significant benefit ($p < 0.01$, using the sign test) in all three conditions when correlation is computed using binary ratings over scalar ones. For the Average, there is a small but insignificant benefit to using binary ratings in computing the average.

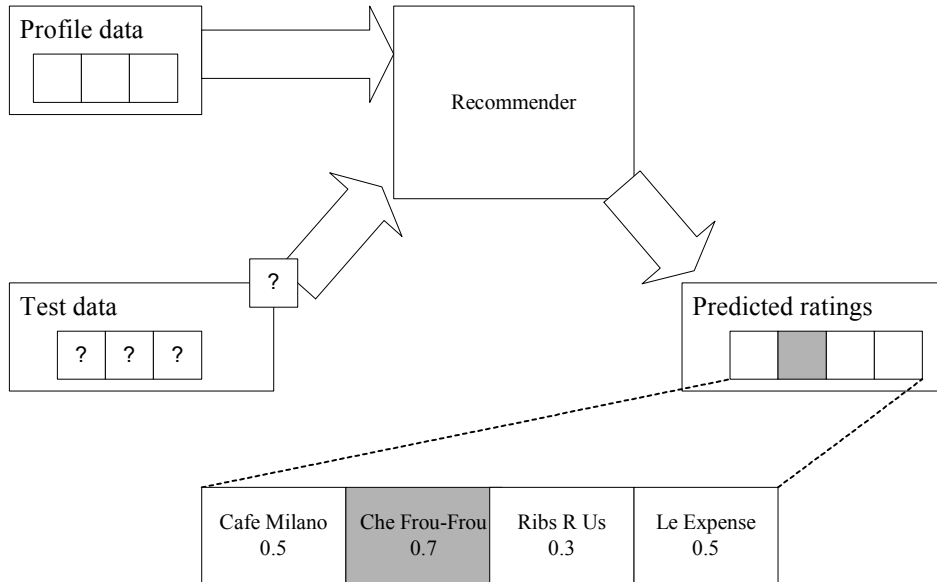


Figure 7. Evaluation Example: Process

These results were surprising in that the scalar ratings were designed to capture some of the differences in meaning between different ratings and yet failed to produce better performance. This most likely indicates that it is a mistake to consider “Browse” to be less negative a critique than the more direct navigation operations. After all, the critique operations indicate that the user found something to like about the current restaurant, whereas browsing away does not. For the remainder of our experiments, only binary ratings were used.

Our next experiment had five conditions: a Random ranking (essentially what Entree does without hybridization), the Average rating of all users, inter-user Correlation, the Sparse metric using cosine similarity and the Heuristic metric. Figure 9 shows the results of this experiment. The Correlation technique outperformed the Sparse version,

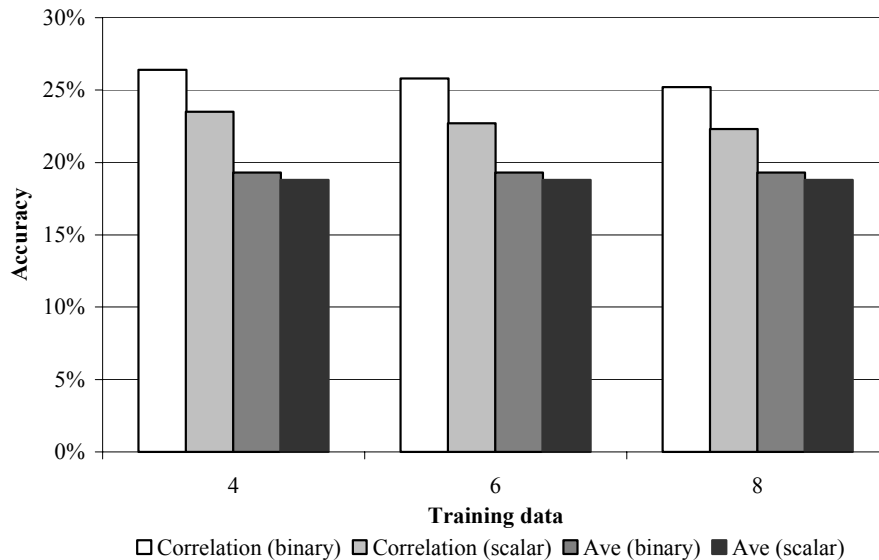


Figure 8. Comparing Binary and Scalar Ratings, Session Size = 15 (5x5 cross-validation, n=4,632)

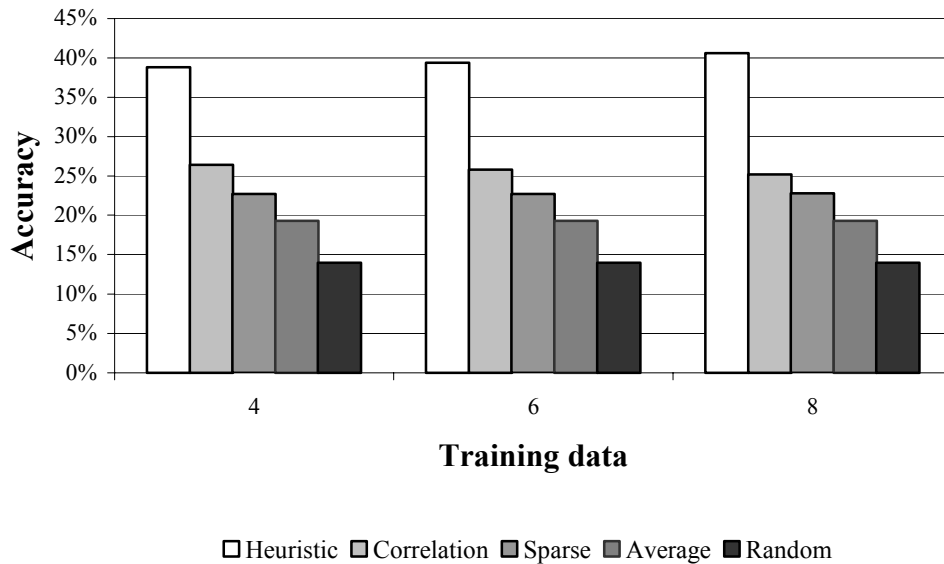


Figure 9. Comparing Collaborative Algorithms, Session Size = 15 (5x5 cross-validation, n=4,632)

but both only outperformed the simple average by single digits. Both techniques' learning curves were essentially flat, indicating that the 15-item profile is not really enough to get the full power of these techniques. The Heuristic technique was the clear winner, achieving 42% accuracy after seeing eight ratings, and even with only four ratings, it had 38% accuracy. The results for all pairs of techniques at each increment of training data were compared using the sign test. All within-condition differences were significant at the $p < 0.01$ level.

Since 15 rating sessions were such a small percentage of the data that we had from Entree, we conducted similar experiments using shorter sessions. Figures 10 and 11 show the results for 10 and 5 item sessions respectively. For

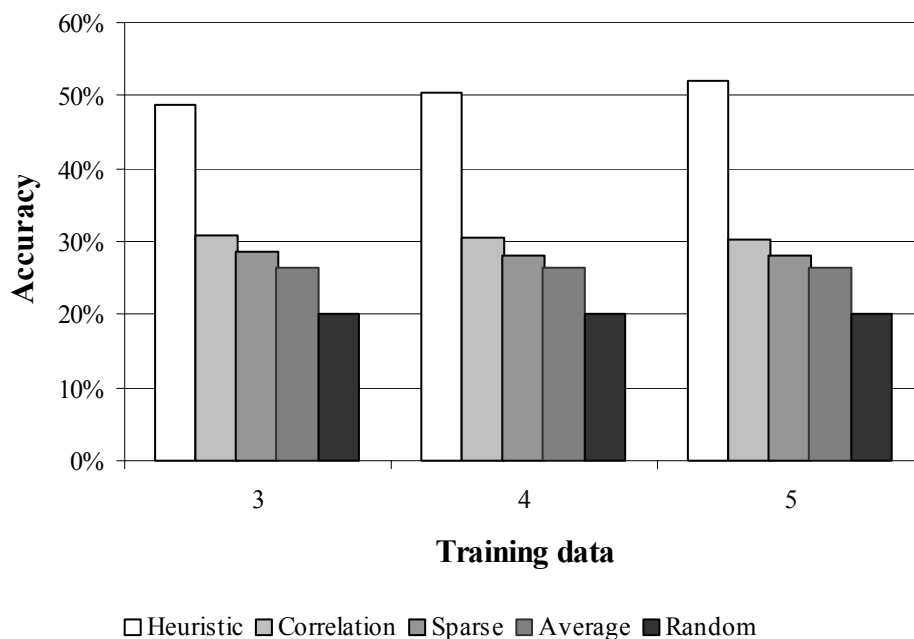


Figure 10. Comparing Collaborative Algorithms, Session Size = 10 (50% sample, 5x5 cross-validation, n=17,030)

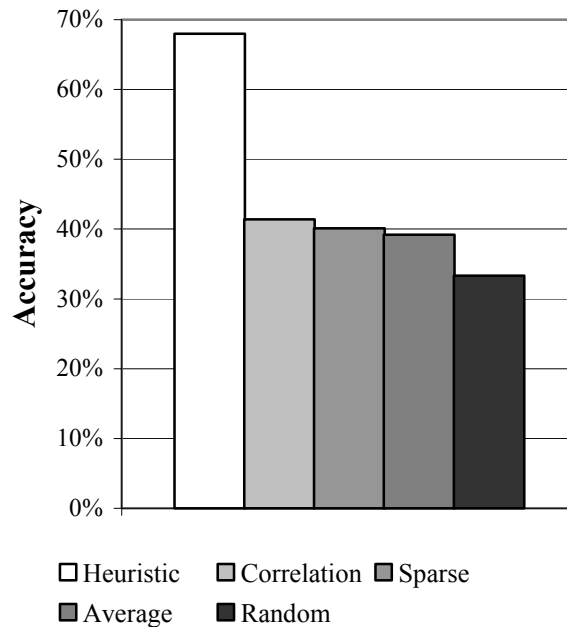


Figure 11. Comparing Collaborative Algorithms, Session Size = 5 (20% sample, 5x5 cross-validation, n=41,697)

these experiments, we sampled the data (50% and 20%, respectively) rather than use every session of the requisite size. The important thing to notice about these conditions is that the baseline is significantly raised — in the 10 rating session, we use five ratings for testing and five as the profile, so a random method is 20% accurate at finding the right answer from the remaining five. In the five rating sessions, two ratings formed the profile and three the test data, so the random baseline is 33%. Still, the relative performance of the techniques is similar to that found in the initial experiment. The other techniques are relatively flat, but the Heuristic technique has a significant advantage that improves with more data. This effect is clearly seen in Figure 12, which plots the boost over the average that each technique achieves for the different size sessions.

6. Discussion

These experiments illustrate a synergy that is possible in the knowledge-based/collaborative hybrid. As with other experiments with a range of hybrid designs, our results confirm the benefits of hybrid recommendation. In addition, because the knowledge-based part of Entree solicits semantic ratings, the collaborative component can take advantage of the extra user knowledge available in these ratings to improve on the standard collaborative technique.

Because the knowledge-based recommender comes first in the cascade, the system does not suffer significantly from the ramp-up problem. If there is no collaborative data to work from, the recommender relies on its knowledge base and selects randomly when items are under-discriminated. The system therefore has an immediate benefit to the casual user, but still improves its recommendations over time as a user’s niche is better defined. What we give up in this hybrid is the ability to make cross-genre discoveries. The employment of collaborative data is circumscribed to discriminating the results of previous retrieval, so it cannot bring in novel suggestions.

The knowledge-based part of the hybrid also helps avoid the stability/plasticity problem. A user who has built up a history of vegetarian choices will still see her results biased in that direction, but only within a particular search context. If the user starts EntreeC with a steakhouse as an example, the most similar restaurants will all be steakhouses, and her fellow vegetarians in her (former) collaborative niche will not have many ratings.

Three directions for future research follow. First would be to examine content-based hybrids involving the Entree base system. The Entree system itself uses the known features of restaurants to reason about similarity. These same features could be used to induce a content-based filter to combine with the knowledge-based recommender.

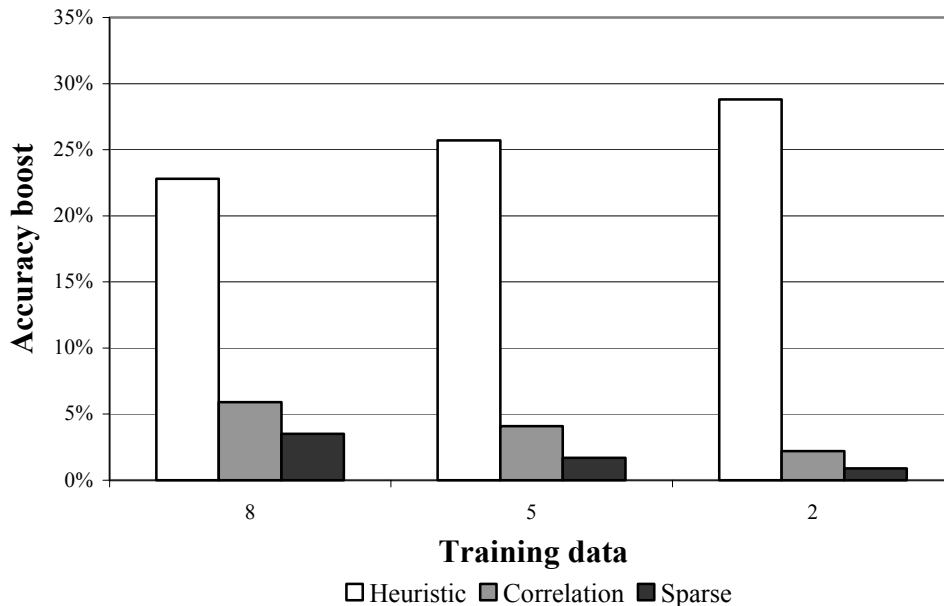


Figure 12. Accuracy Boost over Average

However, the data used in our experiments so far has very short session lengths, and it seems likely that 10 or 15 examples would not be enough to provide good recommendations for any one user.

We can address the short session problem by changing the way that we aggregate data. Our current design simply looks for a temporally-contiguous series of interactions within a short timeframe. Longer (but potentially noisier) sessions could be accumulated by treating the IP addresses of incoming web requests as permanent identifiers, enabling the accumulation of multi-visit profiles. Using long-term sessions will satisfy the data needs of the knowledge-weak recommendation methods, the question is how this change will interact with the very goal-focused design and interface of Entree. Such an experiment will let us determine whether restaurant preferences at the detailed level of navigational critiques are stable over time or if the specific context of each meal tends to override.

Finally, the heuristic method of comparing users is appealing, since it does not lose the semantics of users' choices, but the metric itself must be crafted with these semantics in mind. The metric used in these experiments was developed by enumerating all possible comparisons and exercising simple logical considerations. In effect, we have substituted one type of knowledge engineering (determining the similarity of ratings) for another (developing more finely discriminating similarity metrics). Although the manual crafting of the heuristic similarity metric was successful in this case, this approach is not scalable, especially for domains with more complex navigation options. For the heuristic metric to be useful in other domains, we would need a method of computing or inferring the similarity between ratings.

7. Conclusion

All existing recommender systems employ one or more of a handful of basic techniques: content-based, collaborative, demographic, utility-based and knowledge-based. A survey of these techniques shows that they have complementary advantages and disadvantages. This fact has provided incentive for research in hybrid recommender systems that combine techniques for improved performance. A significant amount of recent research has been dedicated to the exploration of various hybrids, including the six hybridization techniques discussed in this paper: weighted, mixed, switching, feature combination, feature augmentation, and meta-level. A survey of the prominent research in the field indicates that less than half of the 41 possible recommender hybrids have been explored.

Particularly lacking from the literature has been research in knowledge-based recommendation. This article has shown EntreeC, an example of a hybrid recommender system that combines knowledge-based and collaborative techniques using a cascade hybrid. Experiments with EntreeC indicate that collaborative filtering does improve the performance over the knowledge-based component acting alone. Further, the semantic ratings gathered by the

knowledge-based component enabled more accurate prediction of user preference than possible with simple numeric ratings.

Acknowledgements

The EntreeC experiments were designed with the assistance of Daniel Billsus at UC Irvine. Entree was developed at the University of Chicago in collaboration with Kristian Hammond, with the support of the Office of Naval Research under grant F49620-88-D-0058. The interface to the Entree system was designed and created by Robin Hunicke. Many others contributed to the FindMe effort at the University of Chicago, including Terrence Asselin, Kai Martin, Kass Schmitt and Robb Thomas.

References

- Alspector, J., Koicz, A., and Karunanithi, N.: 1997, 'Feature-based and Clique-based User Models for Movie Selection: A Comparative Study'. *User Modeling and User-Adapted Interaction* 7, 279-304.
- Avery, C. and Zeckhauser, R.: 1997, 'Recommender systems for evaluating computer messages'. *Communications of the ACM* 40(3), 88-89.
- Balabanovic, M.: 1998, 'Exploring versus Exploiting when Learning User Models for Text Representation'. *User Modeling and User-Adapted Interaction* 8(1-2), 71-102.
- Balabanovic, M.: 1997, 'An Adaptive Web Page Recommendation Service'. In: *Agents 97: Proceedings of the First International Conference on Autonomous Agents*, Marina Del Rey, CA, pp. 378-385.
- Basu, C., Hirsh, H. and Cohen W.: 1998, 'Recommendation as Classification: Using Social and Content-Based Information in Recommendation'. In: *Proceedings of the 15th National Conference on Artificial Intelligence*, Madison, WI, pp. 714-720.
- Belkin, N. J. and Croft, W. B.: 1992, 'Information Filtering and Information Retrieval: Two Sides of the Same Coin?' *Communications of the ACM* 35(12), 29-38.
- Billsus, D. and Pazzani, M.: 2000. 'User Modeling for Adaptive News Access'. *User-Modeling and User-Adapted Interaction* 10(2-3), 147-180.
- Breese, J. S., Heckerman, D. and Kadie, C.: 1998, 'Empirical analysis of predictive algorithms for collaborative filtering'. In: *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence*, pp. 43-52.
- Brin, S. and Page, L.: 1998, 'The anatomy of a large-scale hypertextual {Web} search engine'. *Computer Networks and ISDN Systems*, 30(1-7), 107-117.
- Burke, R.: 1999a, 'The Wasabi Personal Shopper: A Case-Based Recommender System'. In: *Proceedings of the 11th National Conference on Innovative Applications of Artificial Intelligence*, pp. 844-849.
- Burke, R.: 1999b, 'Integrating Knowledge-Based and Collaborative-Filtering Recommender Systems'. In: *Artificial Intelligence for Electronic Commerce: Papers from the AAAI Workshop (AAAI Technical Report WS-99-0 1)*, pp. 69-72.
- Burke, R.: 2000, 'Knowledge-based Recommender Systems'. In: A. Kent (ed.): *Encyclopedia of Library and Information Systems*. Vol. 69, Supplement 32.
- Burke, R., Hammond, K., and Young, B.: 1997, 'The FindMe Approach to Assisted Browsing'. *IEEE Expert*, 12 (4), 32-40.
- Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D. and Sartin, M.: 1999, 'Combining Content-Based and Collaborative Filters in an Online Newspaper'. *SIGIR '99 Workshop on Recommender Systems: Algorithms and Evaluation*. Berkeley, CA. <URL: http://www.cs.umbc.edu/~ian/sigir99-rec/papers/claypool_m.ps.gz>
- Condliff, M. K., Lewis, D. D., Madigan, D. and Posse, C.: 1999, 'Bayesian Mixed-Effects Models for Recommender Systems'. *SIGIR '99 Workshop on Recommender Systems: Algorithms and Evaluation*. Berkeley, CA. <URL: http://www.cs.umbc.edu/~ian/sigir99-rec/papers/condliff_m.ps.gz>
- Dron, J., Mitchell, R., Siviter, P. and Boyne, C.: 1999, 'CoFIND – an Experiment in N-dimensional Collaborative Filtering'. In: *Proceedings of WebNet '99*. Association for the Advancement of Computing in Education.
- Foltz, P. W.: 1990, 'Using Latent Semantic Indexing for Information Filtering'. In: R. B. Allen (ed.): *Proceedings of the Conference on Office Information Systems*, Cambridge, MA, pp. 40-47.
- Guttman, Robert H.: 1998, 'Merchant Differentiation through Integrative Negotiation in Agent-mediated Electronic Commerce'. Master's Thesis, School of Architecture and Planning, Program in Media Arts and Sciences, Massachusetts Institute of Technology.
- Guttman, R. H., Moukas, A. G. and Maes, P.: 1998, 'Agent-Mediated Electronic Commerce: A Survey'. *Knowledge Engineering Review*, 13 (2), 147-159.

Hill, W., Stead, L., Rosenstein, M. and Furnas, G.: 1995, 'Recommending and evaluating choices in a virtual community of use'. In: *CHI '95: Conference Proceedings on Human Factors in Computing Systems*, Denver, CO, pp. 194-201.

Jennings, A. and Higuchi, H.: 1993, 'A User Model Neural Network for a Personal News Service.' *User Modeling and User-Adapted Interaction*, **3**, 1-25.

Kolodner, J.: 1993, 'Case-Based Reasoning'. San Mateo, CA: Morgan Kaufmann.

Konstan, J. A., Riedl, J., Borchers, A. and Herlocker, J. L.: 1998, 'Recommender Systems: A GroupLens Perspective.' In: *Recommender Systems: Papers from the 1998 Workshop (AAAI Technical Report WS-98-08)*. Menlo Park, CA: AAAI Press, pp. 60-64

Krulwich, B.: 1997, 'Lifestyle Finder: Intelligent User Profiling Using Large-Scale Demographic Data'. *Artificial Intelligence Magazine* **18** (2), 37-45.

Lang, K.: 1995, 'Newsweeder: Learning to filter news'. In: *Proceedings of the 12th International Conference on Machine Learning*, Lake Tahoe, CA, pp. 331-339.

Littlestone, N. and Warmuth, M.: 1994, 'The Weighted Majority Algorithm'. *Information and Computation* **108** (2), 212-261.

Mooney, R. J. and Roy, L.: 1999, 'Content-Based Book Recommending Using Learning for Text Categorization'. *SIGIR '99 Workshop on Recommender Systems: Algorithms and Evaluation*. Berkeley, CA. <URL: http://www.cs.umbc.edu/~ian/sigir99-rec/papers/mooney_r.ps.gz>

Nichols, D. M.: 1997, 'Implicit Rating and Filtering'. In *Proceedings of the Fifth DELOS Workshop on Filtering and Collaborative Filtering*, Budapest, Hungary, pp. 31-36, ERCIM.

Pazzani, M. J.: 1999, 'A Framework for Collaborative, Content-Based and Demographic Filtering'. *Artificial Intelligence Review*, **13** (5/6), 393-408.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P. and Riedl, J.: 1994, 'GroupLens: An Open Architecture for Collaborative Filtering of Netnews'. In: *Proceedings of the Conference on Computer Supported Cooperative Work*, Chapel Hill, NC, pp. 175-186.

Resnick, P. and Varian, H. R.: 1997, 'Recommender Systems'. *Communications of the ACM*, **40** (3), 56-58.

Rich, E.: 1979, 'User Modeling via Stereotypes'. *Cognitive Science* **3**, 329-354.

Richter, M.: 1995, 'The knowledge contained in similarity measures'. Invited talk at the International Conference on Case-Based Reasoning, Sesimbra, Portugal, October 25. Summary available at <URL: <http://www.cbr-web.org/documents/Richtericcbr95remarks.html>>

Rocchio, Jr., J.: 1971, 'Relevance feedback in information retrieval'. In *The SMART System – Experiments in Automatic Document Processing*, New York: Prentice Hall, pp. 313-323.

Rosenstein, M. and Lochbaum, C.: 2000, 'Recommending from Content: Preliminary Results from an E-Commerce Experiment.' In: *Proceedings of CHI'00: Conference on Human Factors in Computing*, The Hague, Netherlands.

Salton, G., and McGill, M.: 1983, 'Introduction to modern information retrieval'. New York: McGraw-Hill.

Sarwar, B. M., Konstan, J. A., Borchers, A., Herlocker, J. Miller, B. and Riedl, J.: 1998, 'Using Filtering Agents to Improve Prediction Quality in the GroupLens Research Collaborative Filtering System'. In: *Proceedings of the ACM 1998 Conference on Computer Supported Cooperative Work*, Seattle, WA, pp. 345-354.

Schafer, J. B., Konstan, J. and Riedl, J.: 1999, 'Recommender Systems in E-Commerce'. In: *EC '99: Proceedings of the First ACM Conference on Electronic Commerce*, Denver, CO, pp. 158-166.

Schmitt, S. and Bergmann, R.: 1999, 'Applying case-based reasoning technology for product selection and customization in electronic commerce environments.' *12th Bled Electronic Commerce Conference*. Bled, Slovenia, June 7-9, 1999.

Schwab, I., Kobsa, A. and Koychev, I.: 2001, 'Learning User Interests through Positive Examples Using Content Analysis and Collaborative Filtering'. *User Modeling and User-Adapted Interaction*. To appear.

Shardanand, U. and Maes, P.: 1995, 'Social Information Filtering: Algorithms for Automating "Word of Mouth"'. In: *CHI '95: Conference Proceedings on Human Factors in Computing Systems*, Denver, CO, pp. 210-217.

Smyth, B. and Cotter, P. : 2000, 'A Personalized TV Listings Service for the Digital TV Age'. *Knowledge-Based Systems* **13**: 53-59.

Strang, G.: 1988, *Linear Algebra and Its Applications*, New York: Harcourt Brace.

Terveen, L. and Hill, W.: 2001, 'Human-Computer Collaboration in Recommender Systems'. In: J. Carroll (ed.): *Human Computer Interaction in the New Millenium*. New York: Addison-Wesley.

Towle, B. and Quinn, C.: 2000, 'Knowledge Based Recommender Systems Using Explicit User Models'. In *Knowledge-Based Electronic Markets, Papers from the AAAI Workshop*, AAAI Technical Report WS-00-04. pp. 74-77. Menlo Park, CA: AAAI Press.

- Tran, T. and Cohen, R.: 2000, 'Hybrid Recommender Systems for Electronic Commerce'. In *Knowledge-Based Electronic Markets, Papers from the AAAI Workshop*, AAAI Technical Report WS-00-04. pp. 78-83. Menlo Park, CA: AAAI Press.
- U.S. Information Technology Industry Council, 'The Protection of Personal Data in Electronic Commerce', *Public Policy Document*, Nov. 20, 1997. <URL: http://www.itic.org/iss_pol/ppdocs/pp-privprin.html >
- Wasfi, A. M.: 1999, 'Collecting User Access Patterns for Building User Profiles and Collaborative Filtering'. In: *IUI '99: Proceedings of the 1999 International Conference on Intelligent User Interfaces*, Redondo Beach, CA, pp. 57-64.
- Zukerman, I. and Albrecht, D.: 2001, 'Predictive Statistical Models for User Modeling'. *User Modeling and User-Adapted Interaction*. To appear.