MOVIES2GO - A new approach to online movie recommendation

Rajatish Mukherjee, Partha Sarathi Dutta, and Sandip Sen
Department of Mathematical & Computer Sciences
University of Tulsa
{rajatish,partha}@euler.mcs.utulsa.edu,
sandip@kolkata.mcs.utulsa.edu

Abstract

We are developing a web-based movie recommender system that catches and reasons with user preferences to pro-actively recommend movies. It combines voting based ranking procedure with guaranteed properties that use syntactic features like actor/actress of movies together with a learning based approach that processes semantic features of movies like its synopsis. Our primary concern is to develop a reasoning procedure that can meaningfully and systematically tradeoff between user preferences. We also provide multiple query modalities by which the user can pose unconstrained, constrained, or instance-based queries. In the paper, we outline the current status of our implementation with particular emphasis on the mechanisms used to provide robust and effective recommendations. We also incorporate some extensions to our previous work including learning modules, explanation facilities, and pro-active information gathering.

Keywords: Recommender system, Voting theory, Learning, User Modeling, Information Agent

1 Introduction

Burgeoning volume of information on the Web has created increasing interest in automated filtering, refinement and organized presentation of relevant information to users. We often need to obtain and refine information from the Internet (a huge and often unstructured source of information) to achieve our goals. There has been efforts to automate filtering of relevant information and presenting organized information (the summary) to the user. Such automated methods, commonly referred to as intelligent information retrieval are used to locate and retrieve information with respect to a user's individual preferences [Balabanovic, 1998; Lang, 1995; M.Pazzani and D.Billsus, 1997; Maglio and Barett, 1997; Thomas and Fischer, 1996]. Also, efforts to rank/sort information based on user preferences, which are often conflicting, has generated interest over the past few years [Schafer *et al.*, 1999a] . One key area of research to achieve this goal is targeted around *recommender systems* (RS). Recommender systems are agent-based systems that use stored user preferences to locate and suggest items of interest to users they serve. They represent a combination of information retrieval and intelligent agent systems. Recommender systems are being used in an ever-increasing number of e-commerce sites to assist "buyers" in finding suitable products [Schafer *et al.*, 1999b]. However, effective recommendation should be able to tradeoff between conflicting user preferences in a meaningful way. Typical application domains for recommender systems include recommendations for music CDs and cassettes [cdnow,], movies [moviecritic,], books, etc. Typically a domain has several features/dimensions. Each dimension has a set of elements and users provide values for their preferences on these elements. For example, in a RS for movies, one dimension may describe the type of a movie, and contain elements like horror, comedy, tragedy, musical, action, etc. A recommender system usually combines the values/ratings of the elements of every dimension according to some evaluation scheme before obtaining a recommendation rating of an item.

Typical evaluation schemes for combining ratings may involve linear or non-linear combinations of feature values. In our movie recommender system, we have used a classical voting method as the evaluation scheme. Voting schemes have been used for ages in political science and economics to select compromise choices from several conflicting alternatives. So, using voting theory in our system is logical because it holds the promise of recommending items that have the desirable properties required to satisfy user preferences. Another key feature of our system is the incorporation of a Bayesian learning mechanism to learn frequently occurring keywords in the synopsis of recommended movies. This ensures improved quality of recommendation and reducing errors that might be introduced by human users. Unlike other recommender systems that use social filtering, our system is based on *individual* user preferences, obtained as user input (initial preferences for movie dimensions) or from interactions with the user (posing queries like: "Did you like this movie?").

In this paper we describe a Web-based movie recommending agent system, with particular emphasis on how our agent stores and uses user preferences to make useful recommendation. We claim that, even in the presence of conflicting user preferences, our movie recommender system, using voting theory-based reasoning scheme, provides satisfactory recommendations.

2 Other Recommender Systems

Many of the technologies used in actual recommender systems are fairly simple database queries. However, a special class of recommender systems, *automatic recommender systems* [Schafer *et al.*, 2001] use a wide range of technologies, ranging from nearest neighbor algorithms to Bayesian analysis [B.M.Sarwar *et al.*, 2000]. Early work in recommender systems show widespread use of nearest neighbor collaborative filtering/social filtering techniques [Shardanand and Maes, 1995; Schafer *et al.*, 1999a]. These techniques are based on the 'word-of-mouth' recommendation relying on the opinion of peers. They function by:

- Recording behavior of a large number of people (e.g. web pages visited, explicit item ratings provided, previous purchases),
- Selecting a number of "neighbors" for the current user (i.e. other people whose past behavior is similar to this person)
- Extrapolate future behavior for the user, based on behavior of these neighbors (e.g. this user will probably like the movie "Entrapment", because many of his neighbors did).

The necessary prerequisite to using such collaborative filtering is that these systems deal with large numbers of people ($>$ 100,000) with many different behavioral aspects ($>$ 1000), few values per user (1 % per user), many recorded values ($>>$ 1,000,000), and have to operate in real-time ($<$ 0.05 seconds). However, such accumulation of data may not be possible, or if possible, may not meet the requirements. Also, searching for neighbors may become extremely slow as the database size increases.

2.1 Related Systems Using Movie Domain

Several systems (using the movie domain) have been developed which attempt to model user preferences. For example, the MORSE [Fisk, 1995; 1997], a movie recommendation system makes personalized movie recommendations based on the principle of social filtering. Other online systems based on the same principle include Firefly (recommends movie, films) [firefly,], Movie Critic [moviecritic,] etc. However, the most popular web based movie recommender system using collaborative filtering is the *Movie Lens* recommender developed by the *GroupLens* [Grouplens,] research group at the University of Minnesota.

However, to the best of our knowledge, none of the available systems use Voting schemes and text-based learning for personalized recommendation. We also introduce the concept of pro-active information gathering which enables the system to become more user-friendly.

3 Research Issues

Consider a recommender system for recommending web sites. We will present various approaches to obtaining user preferences. These approaches can be grouped into two broad categories:

Explicit querying: In these schemes, the user is interactively engaged to collect information about his/her preferences. These approaches can be further divided into two subclasses:

Dimensional listing: In this approach, the user is first asked to list the set of dimensions (features or attributes) of relevance in the domain as well as the domains of those dimensions. For example, in the web page recommendation service some of the dimensions and their domains (set of options in a dimension) are as follows:

Page type: news, sports, entertainment, financial, travel, educational

Source type: commercial, private, governmental, university, charity

Page update rate: monthly, weekly, daily, hourly, per minute

Graphical content: low, medium, high

Access charges: free, sign-up cost, yearly fee, monthly fee, per access cost

The user is expected to input his/her preferences for some of the options for each dimensions. Other options may be assigned default values. Let us assume that preferences are input in a scale of 0 to 1 with more preferred options given higher values. For example, a particular user can rate *news* and *financial* page types highly (say 0.9), and provide a low rating for *sports* page type (say 0.2). The other page types may be provided a default preference (say 0.4). The user will also be asked to rate the relative importance of different dimensions. For example, the above-mentioned user can rate the *page type* and *page update rate* dimensions highly, and provide a low rating for *source type* dimension. Again, dimensions not rated can be provided default values. The system can then use these set of preferences to analyze the relevance of different web pages it accesses. Based on this analysis, web pages will be selected for presentation to the user. A concern for this kind of systems will be that it should not overburden the user by asking for too much information. This is a difficult issue because the more information the system gets, the better the quality of recommendation it can provide to the user. Another major concern is the robustness of the system to errors in user specification of his/her preferences. The user can only be expected to be approximately correct in specifying his/her ratings. Small variations in these values should not significantly alter the recommendations provided by the system.

In our work, we have used the real preference values to obtain an ordinal ranking of the options, and use only these rankings for further analysis. We could also have asked the user to directly input the ordering. But, in our estimate, when the number of options for a dimension is large, providing approximate values on a scale imposes a much lesser cognitive burden on the user than having to sort the list manually according to his/her preferences.

Instance listing: An alternative method for explicitly receiving preference information from the user is to ask him/her for items in the domain he/she did or did not like. For example, in the web page recommendation system, the user may be asked for typical pages that he/she likes and dislikes. These pages can be analyzed to find out patterns or regularities in the likings/dislikings of the user based on underlying dimensions as listed above.

Passive gathering: Passive information gathering is similar in nature to the instance listing approach mentioned above. The primary difference is that instead of asking the user for instances, the system can "watch" the user's browsing habits. Based on frequently visited or book-marked web sites, and sites that the user visits only briefly and does not return to, estimates of user likings/dislikings can be generated.

Related to the above discussion is the question of learning user preferences. It is evident that passive gathering of user preferences amount to learning from observation. Even when the user has been explicitly queried for preferences, it may be necessary to continually update the user model as his/her preferences is apt to change over time. Learning can also be performed by explicit querying or by passive analysis. We believe, that the user would be less inclined to periodically update his/her preferences than he/she would be to input them on first use of the system. As such, we recommend that updating user models be based on passive learning and should be as unobtrusive as possible.

4 System Functionality And Architecture

The recommender system provides the following major functionalities to its users:

- Storing user preferences in the form of the user provided weights of different attributes in several dimensions like favorite actors, favorite actresses, favorite directors, preferred time period (in terms of years), relative likings of different movie genres like comedy, drama, action etc. The system also stores the relative importance of each of these dimensions as specified by the users (e.g. , whether the user rates the genre dimension above the director dimension etc.).
- The system suggests a few most suitable and probably likeable movies. Movies to be suggested are selected by a combination of voting and nearest neighbor algorithms based on the user specific knowledge available to the system. The user can give constrained queries in which he/she can specify values for some of the dimensions (e.g, a user may be looking only for 'comedy' movies directed by a certain director).
- It employs a simple learning mechanism to learn keyword occurrences in the synopsis of movies that a user rates. This makes our system more "individualistic" which provides better recommendation.

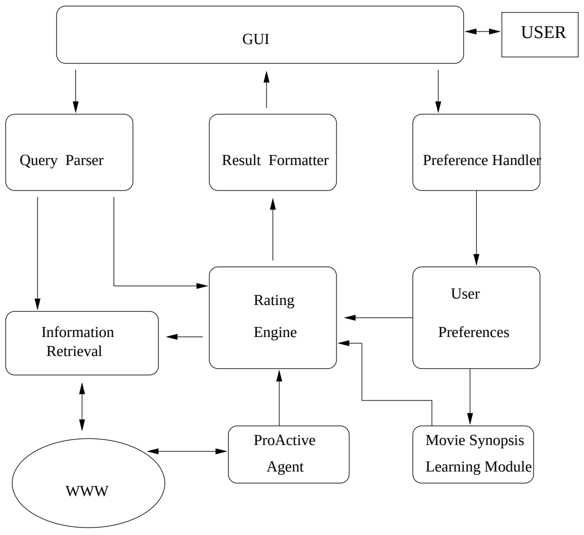


Figure 1: The System Architecture.

The system (refer to Figure 1) in its present form can be viewed in three distinct layers, the graphical user interface (GUI), the main application layer which implements the recommendation strategies (e.g., Voting Engine) and the database layer which consists of the cached movie database and the database of user preferences. An extension of the database layer is the World Wide Web which is used as a virtual repository of information by our system. Currently, the system uses the www.reel.com website as the source of movie information. Given the similarity in querying the online sites, scaling the system to a larger scale may not be difficult.

The Graphical User Interface (GUI), is used for user login, registration (for new users) and obtaining preference information from users. Such information, along with the text-based learning is used to rank the movies. Once a user logs into the system, a movie recommendation page is displayed which is obtained by the system pro-actively using the already set user preferences. The user can also make new queries for movies. Users can make "constrained" or "unconstrained" queries. In a constrained query a user can provide name(s) of actor(s) and/or actresses(es), director, genre, of whom he/she would like a movie. In the current implementation the user can specify upto two actor/actress names, one director and one genre type. An unconstrained query enables a user to simply *find a movie*. Stored preferences for that user are used by our system to retrieve the relevant results. Another option that our system provides is to find a movie similar to a specified movie. To decrease system response time, we cache preference information of old users. We also cache information about the user likings for movies he/she has viewed. These stored details are first looked up by the recommender before searching online.

User queries are directed to the movie database. The application layer ranks the movies it receives from the database according to recommendation schemes. A select list of movies are returned to the user as the recommended movies.

Our movie recommendation system can be accessed at the following URL:
`http://www.mcs.utulsa.edu/˜rajatish/recommender/login.html.`

5 Using User Preferences

For intelligent agent applications to be used by humans, it is essential that these software entities be able to encode and follow the priorities, biases, and preferences of associated users. So, as developers of such systems, we must pay careful attention to effectively representing and utilizing user preferences.

We are careful about limiting the cognitive load imposed on the user of such a system. We provide the user with default preference values for all preference dimensions. We anticipate that some users will not take the time and effort to change all the preferences; providing default preferences that are found to be suitable for a wide range of users allows for effective system performance for this class of users. In future, we plan to investigate learning user preferences from querying other agents. This will allow for a better tuning of system performance.

5.1 Storing User Preferences

We now discuss how we store user preferences. In the following, we use dimensions to refer to user preferences for actors, directors, movie genres, etc. The term option refers to one of the allowable values for a dimension.

A: Actors Lead and supporting actors in a movie.

B: Actresses Lead and supporting actresses in a movie.

C: Director Director of a movie.

D: Type Movie type or genre.

For example, user A may find a movie because actress X is in the cast, but not if the movie genre was drama. Each user assigns a value between 0 and 1 for options of each dimension. Options not rated are provided default values. For example, in Figure 2, a user has specified his/her preferences for movie types. The type 'Documentary' is left unpreferred and so default value of 0.0.

The user is also responsible for rating each dimension of the movie domain. Thus some of these options are given greater weight than others, e.g., the actress of a movie may be more of a deciding factor than the director of the movie.

We have adapted methods developed in the voting theory literature to find compromises between possibly disparate preferences. Voting is a well understood mechanism for reaching consensus [Ordeshook, 1995; Straffin, 1980]. One of our major concerns is that we effectively capture the interactions between the many and conflicting preferences that the user has for a movie. Voting schemes have been successfully used for ages and different contexts to arrive at a compromise choice between several candidates. In general, no one choice appears to be a clear winner along all dimensions under consideration. Voting theory provides some guarantees regarding the compromise solution it provides. With these guarantees the recommender system can generate a convincing argument for recommending a particular movie when its associated user asks for an explanation. The capability to provide a formal explanation of agent behavior has been an added inceptive for the use of voting techniques in our implementation.

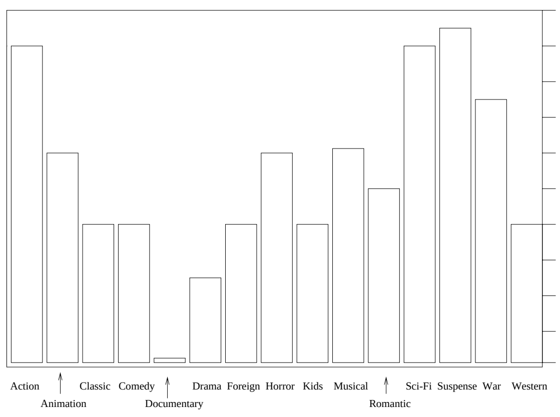


Figure 2: Example preferences for movie types. The default preference is 0.0.

We can assign votes to each preference in proportion to its weight with respect to other preferences. Each movie can then be voted on by the preferences as to whether the user wishes to see it given the particular values for that movie's dimensions. The top few vote getters will be returned as recommendations by the system.

Because stated user preferences may be different than their true preferences, an agent utilizing a voting-based consensus reaching mechanism is not guaranteed to always select movies that will be of interest to the user. In practice, such guarantees are perhaps impossible to provide, and it is very likely that most users will be happy with recommendations that fit their preferences most of the time.

To provide this, our mechanism would have to have two properties:

Effectiveness: Assuming that the stated preferences are correct, the system should recommend movies chosen meaningfully using such preferences.

Robustness: The decision procedure should not be too sensitive to the exact values of stated preferences. For example, minor changes in the preferences should not produce radical changes in the recommendations produced.

We want our selection procedure to scale well as the number of preferences are increased. Additionally, we want the procedure to return only a few recommendations. The voting scheme used to provide recommendation is discussed below.

5.2 Voting Schemes

Given a list of alternatives to choose from $A = \{a_1, a_2, \ldots, a_i\}$, a set of voters $V = \{v_1, v_2, \ldots, v_j$, and a preference function, P that returns the rank ordering of the alternatives given a voter, a voting scheme will produce a ranking of the alternatives. A pairwise voting procedure produces a ranking based on votings over all possible pairings of alternatives, where each pair is voted on by ignoring all other alternatives. Straffin has listed several criterion to rate the desirability of the outcome different voting rules [Straffin, 1980]:

Pareto If every voter prefers an alternative x to an alternative y, a voting rule should not produce y as a winner.

Condercet Winner If there is an alternative x which could obtain a majority of votes in pairwise contests against every other alternative, a voting rule should choose x as the winner.

Condercet Loser If an alternative y would lose in pairwise majority contests against every other alternative, a voting rule should not choose y as a winner.

Monotonicity If x is a winner under a voting rule, and one or more voters change their preferences in a way favorable to x (without changing the order in which they prefer any other alternatives), then x should still be the winner.

Majority If a majority of voters have an alternative x as their first choice, a voting rule should choose x.

Smith's Generalized Condercet If the alternatives can be partitioned into two sets A and B such that every alternative in A beats every alternative in B in pairwise contests, then the voting rule should not select an alternative in B.

Following our requirements for a decision mechanism in the previous section, we conclude that we should use a voting mechanism whose outcome satisfies most or all of the above-mentioned criteria. As such we have chosen Black's voting rule: if there is a Condercet Winner, then choose it, else apply the Borda count voting rule.

The Borda count voting rule works as follows: assign points to an alternative based on its position in a voter's preference list. The last place alternative gets 0 points, the second to last 1 point, and so on until the first place which gets n points. If a voter is indifferent to 2 or more alternatives, then each one is assigned the average of the alternatives. For example, if a voter has the preference list of $p = \{a, \{b, c, d\}, e, \{f, g\}\}$, then the points are awarded as follows: g and f each get 0.5; e gets 2; d, c, and b each get 4; and, a gets 6. The alternative that receives the highest number of votes from all voters is the Borda count winner.

In our scheme, after each dimension is used to provide a Borda count for each movie, these ranks are combined. A simple summation of the rank could have been used if all dimensions were of equal importance to the user. Since, the dimensions have varying weights, we represent each dimension by a number of voters proportional to its weightage. This voter number is calculated as follows for the jth dimension:

$$V_j = \frac{w_j}{\sum_{j \in D} w_j} * 1000,$$

where D is the set of dimensions, and w_j is the weight of the jth dimension. Then, the total vote received by an alternative a_i is given by

$$TV(a_i) = \sum_{j \in D} V_j * r_{ij},$$

where $r_i j$ is the Borda count of alternative a_i according to the jth dimension.

We will now clarify how our selection procedure addresses the robustness criterion mentioned in the last section. In our system, the users input real values in the range $[0, 1]$ for their preferences for some of the options for each dimensions. Options not rated by the user are assigned some different preference values. The system then uses these numbers to only produce an ordinal ranking of each movie for each dimension. For example, if actress A is rated higher than actress B, and movie X and Y includes A and B respectively in their casts, the voters corresponding to the movie dimension will rate X above Y. So, the order of preferring A and B is important in our scheme, not the amount by which one is preferred over the other. As such, small differences in preference estimates for the options will not affect the outcome of the voting scheme.

To compare the effectiveness and robustness of our voting scheme, we have also implemented a straightforward *proportional ranking* method. In this method, the worth of alternative a_i is calculated as:

$$PR(a_i) = \sum_{j \in D} w_j * a_{ij},$$

where D is the set of dimensions, a_{ij} is the option value of the jth dimension for alternative a_i, and w_j is the weight of the jth dimension. One major weakness of this method is its susceptibility to small variations in the stated preferences.

Voting systems have previously been applied to multi-agent system domains [Ephrati and Rosenschein, 1991; Ephrati *et al.*, 1994; Rosenschein, 1995; Ephrati and Rosenschein, 1996] but not in the context of cooperative voters. Ephrati *et al.*'s approach focuses on voting by all users of a meeting to reach consensus on an acceptable time for the meeting and is a non–cooperative voting scenario where voters can vote insincerely [Ephrati *et al.*, 1994]. They can therefore use only those voting strategies that cannot be sabotaged by strategic manipulation [Ordeshook, 1995; Straffin, 1980]. However our approach is for different preferences of the same user to vote on what is a preferable movie, and hence is a cooperative voting system. Thus even though the Borda count is vulnerable to strategic manipulation [Brams and Fishburn, 1991], we need not concern ourselves with this issue as the user has no incentive to artificially and deliberately inflate or deflate his/her preferences for dimensions and corresponding options.

6 Other Features of the Work

There are several other features of our system which makes our approach distinctly different from other recommendation systems. Some of them include:

Learning semantic recommendations: In addition to the syntactic, voting based recommendation, we decided to incorporate a semantic recommendation component that can analyze the contents of movie summaries or synopsis. We have used a naive Bayesian learner to learn text-patterns or keywords of movie synopsis that reflects the preferences of the user for movie contents based on previously rated movies by a user. We have accommodated such keyword learning process into our system recommendation by adding a new dimension to our ranking engine. We use a well-known algorithm to classify text, based on the naive Bayes classifier [Lang, 1995; Mitchell, 1997; M.Pazzani and D.Billsus, 1997]. The basic idea is to observe the frequencies of occurrences of words in the synopsis of rated movies. Then the rating of a new movie is based on the content of its synopsis. The more a movie synopsis contains words that occur more frequently in liked movies, the more the chance that this particular movie will be liked. Probabilistic approaches, such as the one described here are among the most effective algorithms currently known for learning to classify text documents. The Naive Bayes learner makes the learner makes the simplifying assumption that the probability that the probability of the entire synopsis given that the movie has a particular rating is the same as the product of the individual probabilities of the synopsis containing each of the individual words given the movie is of the same rating:

$$\begin{aligned} v_{NB} &= \arg \max_{v_j \in \{like, ok, dislike\}} P(v_j) P(w_1, w_2, \ldots, w_n | v_j) \\ &= \arg \max_{v_j \in \{like, ok, dislike\}} P(v_j) \prod P(a_i | v_j), \end{aligned}$$

where $w_1, \ldots, w_n$ are the n words in the synopsis. This enables the storage of an exponentially fewer frequency counts to derive the required categorization of unseen movies. The effectiveness of this learning mechanism increases with the amount of training data available. This means that as the user uses the system and rates more movies, the system is likely to infer more accurate recommendations. It is also possible to decay the frequency counts over time to track changing user preferences.

The learning problem can be stated as follows: let an instance space X consist of all possible text documents. We are given training examples of some unknown target function $f(x)$, which can take any value from a finite set V. The task is to learn from these training examples to predict the target value for subsequent text documents. For example, in our text-based learning of the movie synopsis, we will consider the target function classifying documents as interesting, ok and uninteresting to a particular person, using the target values *like*, *OK* and *dislike* to indicate these three classes.

To calculate the conditional probabilities, we use the m-estimate - with uniform priors and with m equal to the size of the word vocabulary. Thus, the estimate of $P(w_k|v_j)$ will be $(n_k + 1)/(n + |Vocabulary|)$, where n is the total number of word positions in the training examples whose target value is v_j, n_k is the number of times w_k is found among these n word positions, and $|Vocabulary|$ is the total number of distinct words (and other tokens) found within the training data. Initially, when the system is first uploaded and has no users, the vocabulary is zero. However, with time, when a old user logs in, he/she is asked to rate his last seen movie (based on the system recommendation). Based on the rating, *like*, *OK* and *dislike* which forms the elements of the set V, the probability of the occurrence of words (found in the movie synopsis) are updated in our word database. Also, if a word does not exist in the vocabulary table, it is added. Finally, if not present, the word is added to the user recommended class (class belongs to V) otherwise the count of the number of times the word is encountered in that class is increased by one. Now, when we recommend a movie, we check the number of times the user has used the system, and if it exceeds a predefined value set by us, we use the learning mechanism to classify the synopsis of the movies being recommended. It adds a new dimension to our Voting engine. The weight of this dimension increases with time, being 0 initially till it reaches a maximum value set by the user. This feature will only be useful if the user plans to use the system for a length of time.

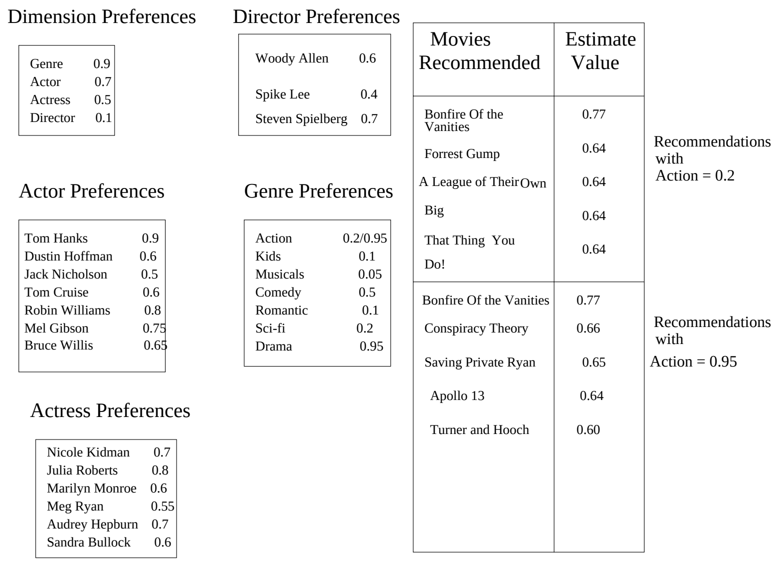
Pro-Active Information Gathering: An important aspect of agent based systems is pro-activeness [Decker *et al.*, 1997]. This feature allows us to recommend to the user newly released movies based on their previous interactions. Every time an old user logs in, he/she is provided with a short list of recommended new releases based on pro-active information gathering.

7 Evaluation of the System

Our movie recommendation system (Movies2Go) currently queries the *www.reel.com* website for the movie information. However, we also cache movie information to decrease the response time. The two principal ways of recommending are:

- Recommendations based on the stored user preferences (unconstrained queries).
- Recommendations based on constraints specified by the user on one or more dimensions and the stored user preferences (constrained queries).

In the following paragraphs, we shall present our analysis of some of the results that we have obtained. Consider the preference values and the recommended movies for the unconstrained query “Find me a movie” (see Figure 3). The previously set value of ‘Action’ by the user is 0.2. The recommended movies are designated as “Recommendations with Action = 0.2”. The *estimated value* of a movie to be recommended is calculated as the ratio of the number of votes it gets to the maximum number of votes a movie can get for the given user preferences. According to the preferences in this example, the ‘Genre’ dimension has the highest value and hence will have the largest number of votes. Within this dimension, the ‘Drama’ option has the highest preference. This explains why the movies recommended have drama as genre. Now let us analyze why *Bonfire of the Vanities* is rated the



Dimension Preferences

Genre	0.9
Actor	0.7
Actress	0.5
Director	0.1

Director Preferences

Woody Allen	0.6
Spike Lee	0.4
Steven Spielberg	0.7

Actor Preferences

Tom Hanks	0.9
Dustin Hoffman	0.6
Jack Nicholson	0.5
Tom Cruise	0.6
Robin Williams	0.8
Mel Gibson	0.75
Bruce Willis	0.65

Genre Preferences

Action	0.2/0.95
Kids	0.1
Musicals	0.05
Comedy	0.5
Romantic	0.1
Sci-fi	0.2
Drama	0.95

Actress Preferences

Nicole Kidman	0.7
Julia Roberts	0.8
Marilyn Monroe	0.6
Meg Ryan	0.55
Audrey Hepburn	0.7
Sandra Bullock	0.6

Movies Recommended	Estimate Value	
Bonfire Of the Vanities	0.77	Recommendations with Action = 0.2
Forrest Gump	0.64	
A League of Their Own	0.64	
Big	0.64	
That Thing You Do!	0.64	
Bonfire Of the Vanities	0.77	Recommendations with Action = 0.95
Conspiracy Theory	0.66	
Saving Private Ryan	0.65	
Apollo 13	0.64	
Turner and Hooch	0.60	

Figure 3: Sample User preferences for Unconstrained Query and corresponding results.

highest. The genres of this movie are ‘Drama’ and ‘Comedy’; its cast includes *Tom Hanks* and *Mel Gibson* who have received high preference from the user. As the genre and actor dimensions have highest weightage relative to other domains, *Bonfire of the Vanities* is rated highest. The remaining movies all cast *Tom Hanks* as the actor (rated first in the actor dimension) and has ‘Drama’ and ‘Comedy’ as the genres (rated first and second in the genre dimension), and hence are recommended.

If the user now changes his/her preference of ‘Action’ movies to be equal to that of ‘Drama’ movies (see Figure 3), movies of type ‘Action’ and casting either *Tom Hanks* or *Julia Roberts* gets higher Borda count, e.g. *Conspiracy Theory, Saving Private Ryan, Apollo 13* etc (see figure 3 under “Recommendations with Action = 0.95”), and they replace the ’Drama’ movies (*Forrest Gump, A League of Their Own, Big,* etc) recommended before. However, *Bonfire of the Vanities* still remains the winner. This result shows that the system is adaptive to user preferences.

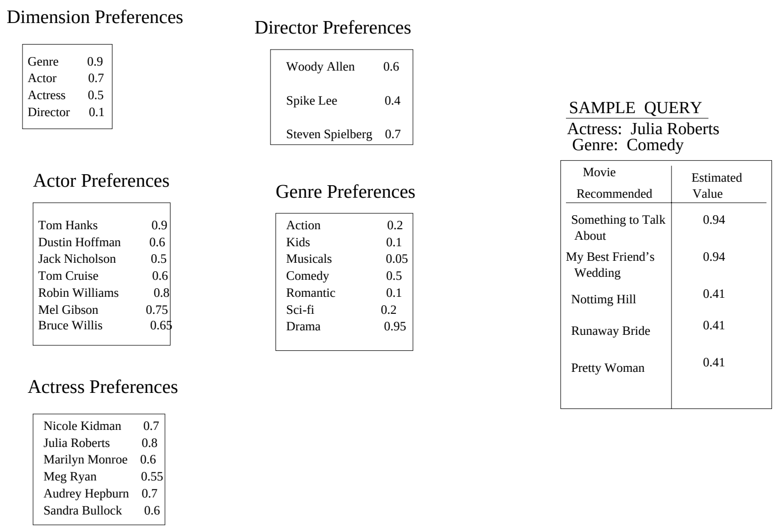
In the case of constrained queries, on the other hand the user gives constraints on one or more dimensions, for example the user might ask the RS to recommend movies of an actor say, Julia Roberts. The user specified constraints filter out the movies in the database and the top movies are recommended according to his/her preferences. One such constrained query for a movie is shown in figure 4. Because of high weightage given to ‘genre’ and ‘actor’ dimensions, *Something to Talk About* turns out to be the winner. Its genres are Drama and Comedy. *My Best Friend’s Wedding*, also being a comedy-drama movie gets the next highest number of votes. *Notting Hill*, a comedy-romance movie is also recommended, but receives fewer votes because the user gave lower preference values for ‘comedy’ and ‘romance’.

As we had discussed earlier, the voting mechanism we propose here is robust to small variations in the user preferences whereas proportional ranking mechanism is not. We changed the user preference for *Mel Gibson* from 0.75 (see Figure 3) to 0.7. Our voting mechanism returns the same list of movies as in Figure 3. However, as an effect of this minor variation, *Conspiracy Theory*, an action movie and casting *Julia Roberts* and *Mel Gibson* is not recommended by the proportional ranking mechanism. In this mechanism, dimension value together with the vote for that dimension decides the final vote for the movie. Since vote for the dimension is high, even small variations have pronounced effect. On the other hand, for the voting scheme using Black’s rule, the relative rankings and the dimension votes are used to calculate the number of votes for a movie. Therefore, as long as the relative rankings within the dimensions remain the same, variations in the preferences will not change the votes for the movie, making the system robust.

8 Conclusion and Future Work

In this paper, we have presented the internal details and the query modalities of a movie recommender system. In particular, we have stressed on the voting based movie selection mechanism that uses stored user preferences for different movie dimensions and options for each dimension. The voting scheme used provides desirable guarantees about the nature of recommendation produced and is also robust to minor variation in specified preferences. This is highly desirable as values from the user are likely to be only approximately correct. We also show the vulnerability of a straightforward proportional ranking based movie selection scheme. There are several ways in which our movie recommendation system can be enhanced. Some of the features that we plan to add are the following:

Learning: We are evaluating the possibility of incorporating other learning schemes by which the system can effec-



Dimension Preferences

Genre	0.9
Actor	0.7
Actress	0.5
Director	0.1

Director Preferences

Woody Allen	0.6
Spike Lee	0.4
Steven Spielberg	0.7

Actor Preferences

Tom Hanks	0.9
Dustin Hoffman	0.6
Jack Nicholson	0.5
Tom Cruise	0.6
Robin Williams	0.8
Mel Gibson	0.75
Bruce Willis	0.65

Genre Preferences

Action	0.2
Kids	0.1
Musicals	0.05
Comedy	0.5
Romantic	0.1
Sci-fi	0.2
Drama	0.95

Actress Preferences

Nicole Kidman	0.7
Julia Roberts	0.8
Marilyn Monroe	0.6
Meg Ryan	0.55
Audrey Hepburn	0.7
Sandra Bullock	0.6

SAMPLE QUERY
Actress: Julia Roberts
Genre: Comedy

Movie Recommended	Estimated Value
Something to Talk About	0.94
My Best Friend's Wedding	0.94
Nottimg Hill	0.41
Runaway Bride	0.41
Pretty Woman	0.41

Figure 4: A constrained Query: Find me a Julia Roberts Comedy movie.

tively update the stored user preferences based on recommendations that were liked or disliked by the user.

Explanation facility: We plan to implement an explanation facility by which the user can be given more details as to why a certain movie was recommended based on his/her preferences. This can also be used by the user to correct or update stored preferences.

The approach that we have proposed in this paper has several shortcomings. For example, though directors in general may have less influence on any user's movie choice, particular directors may by themselves draw movie-goers. One option of accommodating such particularities would be to consider exception-handling rules which will be matched before using the voting scheme. A reasonable exception-handling rule can be to multiply the weight of a dimension by a constant factor if for a given movie the corresponding option ranks within the top few choices for that dimension. Another approach may be to consider new dimensions that combine underline dimensions in some particular manner. This could be an area of active research in the future. Finally, the voting scheme that we have proposed in this paper is not limited to just movie recommender systems. Infact, it has widespread applicability whenever we want to trade-off between disparate and often ambiguous preferences.

References

[Balabanovic, 1998] M. Balabanovic. Learning to surf: Multiagent systems for adaptive web page recommendation. *Ph.D Thesis*, 1998.

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

[Brams and Fishburn, 1991] Steven J. Brams and Peter C. Fishburn. Alternative voting systems. In L. Sandy Maisel, editor, *Political Parties and Elections in the United States: An Encyclopedia*, volume 1, pages 23–31. Garland, New York, 1991.

[cdnow,] Cdnow.com. URL:http://www.CDnow.com/.

[Decker *et al.*, 1997] K. Decker, A. Pannu, K. Sycara, and M. Williamson. Designing behaviors for information agents. In *Proceedings of the 1st International Conference on Autonomous Agents*, pages pages 404–413, Marina del Rey, February 1997.

[Ephrati and Rosenschein, 1991] Eithan Ephrati and Jeffrey S. Rosenschein. The Clarke Tax as a consensus mechanism amoung automated agents. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 173–178, Anaheim, CA, July 1991.

[Ephrati and Rosenschein, 1996] Eithan Ephrati and Jeffrey S. Rosenschein. Deriving consensus in multiagent systems. *Artificial Intelligence*, 87(1-2):21–74, November 1996.

[Ephrati *et al.*, 1994] Eithan Ephrati, Gilad Zlotkin, and Jeffrey S. Rosenschein. A non-manipulable meeting scheduling system. In *13th International Workshop on Distributed Artificial Intelligence*, July 1994.

[firefly,] Firefly.com. URL:http://www.firefly.com/.

[Fisk, 1995] D. Fisk. Recommending films using social filtering. *BT MSc Dissertation*, 1995.

[Fisk, 1997] D. Fisk. An application of social filtering to movie recommendation. *Software agents and soft computing. Towards enhancing machine intelligence. Concepts and applications, Berlin, Germany*, pages 116–131, 1997.

[Grouplens,] Movie lens. URL:http://www.cs.umn.edu/Research/GroupLe

[Lang, 1995] K Lang. Newsweeder: Learning to filter news. In *Proceedings of the 12th International conference on machine learning*, pages 331–339. Morgan Kaufmann, San Fransisco, 1995.

[Maglio and Barett, 1997] P. Maglio and R. Barett. How to build modeling agents to support web searches. In *Proceedings of the Sixth International Conference on User Modeling*, pages 5–16, 1997.

[Mitchell, 1997] Tom M. Mitchell. *Machine Learning*. Mc Graw Hill, 1997.

[moviecritic,] Moviecritic.com. URL:http://www.moviecritic.com/.

[M.Pazzani and D.Billsus, 1997] M.Pazzani and D.Billsus. Learning and revising user profiles: the identification of interesting web sites. *Machine Learning*, pages 313–331, 1997.

[Ordeshook, 1995] Peter C. Ordeshook. *Game Theory and Political Theory: An Introduction*. Cambridge University Press, 1995.

[Rosenschein, 1995] Jeffrey S. Rosenschein. Multiagent planning as a social process: Voting, privacy, and manipulation. In Victor Lesser, editor, *Proceedings of the First International Conference on Multi–Agent Systems*, page 431, San Francisco, CA, 1995. MIT Press. (invited speaker talk).

[Schafer *et al.*, 1999a] J. Ben Schafer, Nathaniel Good, and Joseph Konstan et. al. Combining collaborative filtering with personal agents for better recommendations. In *Proceedings of the 1999 National Conference of the American Association of Artificial Intelligence*, pages 439–436, 1999.

[Schafer *et al.*, 1999b] J. Ben Schafer, Joseph Konstan, and John Riedl. Recommender systems in e-commerce. In *ACM Conference on Electronic Commerce (EC-99), Denver, CO*, November 1999.

[Schafer *et al.*, 2001] J.B. Schafer, J.Konstan, and J.Riedl. Electronic commerce recommender applications. *Journal of Data Mining and Knowledge Discovery*, 5:115–152, 2001.

[Shardanand and Maes, 1995] Upendra Shardanand and Pattie Maes. Social information filtering: Algorithms for automating word of mouth. In *Proceedings of the conference on Computer supported cooperative work*, pages 210–217, 1995.

[Straffin, 1980] Philip D. Jr. Straffin. *Topics in the theory of voting*. The UMAP expository monograph series. Birkhauser, Boston, MA, 1980.

[Thomas and Fischer, 1996] C. Thomas and G. Fischer. Using agents to improve the usability and usefulness of the world wide web. In *Proceedings of the Fifth International Conference on User Modeling*, pages 5–12, 1996.