Semantic feedback for hybrid recommendations in Recommendz

Matthew Garden and Gregory Dudek
McGill University
Centre For Intelligent Machines
3480 University St, Montréal, Québec, Canada H3A 2A7
{mgarden, dudek}@cim.mcgill.ca

Abstract

In this paper we discuss the Recommendz [1] recommender system. This domain-independent system combines the advantages of collaborative and content-based filtering in a novel way. By allowing users to provide feedback not only about an item as a whole, but also properties of an item that motivated their opinion, increased performance seems to be achieved. The features used to describe items are specified by the users of the system rather than predetermined using manual knowledge-engineering. We describe a method for combining descriptive features and simple ratings, and provide a performance analysis.

1. Introduction

This paper describes an approach to the design of recommender systems [16] that provides both higher accuracy and richer feedback than alternative methods. This work is exemplified in a system we have developed called *Recommendz*. A recommender system is a mechanism providing suggestions regarding items of interest based on knowledge of a user's tastes. Most recommender systems are web-based and base their suggestions on knowledge of a user's existing tastes in the domain of interest (for example, what movies they like and dislike). Our recommender system is web-based and contains a database of items from multiple domains of interest (such as movies). Visitors to the site provide feedback on particular items (such as movies they have seen), and are then provided with predicted ratings on items they have not yet rated.

Existing recommendation systems exploit both similarities between users of the system and similarities in the "item structure", but most techniques use only a single scalar measurement per item for each user (i.e. "did this user buy this item", or "did this user like this item"). Recommendz differs in the type of feedback collected from the users as well as in the algorithm that computes the recommendation with this feedback. Broadly speaking, collaborative filtering refers to making recommendations by matching users to other users, and then exploiting a transitive relationship between users and the items they rate: *User A is similar to User B, and User B likes item C, therefore A will like Item C*. An alternative methodology for recommendation systems is to employ content (or item) based filtering, directly matching items without explicitly referring to other users: *User A likes Item D, Item D is similar to Item C, therefore User A will like Item C*. In either case, the similarity relationship between users or items is crucial, yet must be inferred based on very sparse data.

Our approach is a hybrid of traditional collaborative filtering and content-based filtering; however, our content information comes from the ratings of users, rather than from a pre-existing database of item information or from an automated analysis of the items in consideration (although it is possible that such information could be profitably incorporated into our system).

It is common in recommender systems for the user to provide feedback about an item as a whole in the form of a numeric rating in some range (e.g. from 1 to 5). In our system the user gives feedback in this form, but is also required to provide feedback on at least one *feature* of the item being rated. These features can be positive or negative attributes. Specifically, the user specifies the quantity of the feature associated with the item on a numeric scale, and then rates the feature as being a negative or positive aspect of the item, also on a numeric scale. For example, the film "Star Wars" is frequently rated as being associated with a fairly high level (or quantity) of the feature "Robots", and users vary in the extent to which they like or dislike this attribute of the film.

There are several ways this semantic information can be exploited to provide recommendations. Typically, the similarity between a pair of users is calculated based on whether they have rated a similar set of items, or on how similar the

[1] http://www.recommendz.com

numerical ratings for common items have been. The feature information our system learns provides a sense of *why* the users have preferences for certain items over others and can help define the causative or explanatory basis for the ratings. This allows the system both to gain a better understanding of the similarity between users and to (better) explain the recommendations made. Other researchers have identified good explanations as being an important aspect of recommender systems, in that they build user trust [13].

Another use of feature information is in building profiles about the content of items. Such profiles may be useful to users, but they can also be exploited in content-based algorithmic variants (an issue we are examining, but which is outside the scope of this paper).

In most non-trivial domains, the set of important features is much smaller than the set of all possible items; this appears to be the case in the movie domain, as one might expect. This means that even in cases where traditional collaborative filtering is not able to determine a similarity between two users because no items were rated in common, it is still possible that they have used features in common, and on that basis we can make recommendations.

Although the space of features is, at least in the domain under consideration, smaller than the space of items, it is still large enough that making the entire list available to each user when rating would be overwhelming. For that reason we use a statistical method to determine similar features based on usage, and to suggest, for a given item, a set of features which may be appropriate for the user and item.

2 Background

The system called Tapestry is often associated with the genesis of computer-based recommendation and collaborative filtering systems. In Tapestry [10], users were able to annotate documents with text comments. Other users were then able to query these annotations to find suitable documents. Both the annotation and filtering process in this system were performed manually. The key attribute of this system is that it allowed recommendations to be generated based on a synthesis of the input from many other users.

The collaborative paradigm begun with Tapestry was later automated in a number of projects [16, 3, 14].

Recommender systems can be broadly classified with respect to two algorithmic variations: pure collaborative filters and content-based filters. Collaborative filtering is the process of filtering based on preference data from a community of users, ignoring characteristics of the items in question. Similarity between users is determined based on preference data, and then preferred items are recommended to similar users. In a variation on this approach, it has been shown that it can be useful to calculate the similarity between items based on preference data, and then to recommend similar items to the user [17]. Content-based filtering is the opposite of pure collaborative filtering: User interests are captured in some way, content profiles are learned for items, and users are then matched to items based on the degree to which interests and content coincide.

In the Entree system [4, 5], Burke created a recommender system for restaurants in which user feedback is not specified as a numeric overall rating, but in which users specify a semantic rating in which he or she feels the current item is lacking. The system uses the feedback both to learn what the user is looking for and to build a profile of each item. It is then possible to recommend restaurants which fit the user's desires. In Entree, the set of semantic ratings available is predetermined by manual knowledge engineering.

The CoFIND [8] system also uses a semantic rating system. In this system, the user provides feedback on the "qualities" of the item and qualities are suggested if they are used repeatedly. The emphasis in CoFIND is in organizing resources to aid in learning.

The problem of how to efficiently prompt the user to provide useful feedback, known as *preference elicitation*, is important to recommender systems [1]. The issue is to identify those items which will be useful to the system in predicting preferences, so that the user can provide as much information with as little burden as possible. In our case we wish to identify items and features which are actually useful. Several researchers have examined approaches to this problem, including Partially Observable Markov Decision Processes [1] and the Expected Value of Information [2]. For a theoretical analysis of this problem, see [7].

One straightforward method of making recommendations is to find the nearest neighbors of a particular user, according to some similarity measure [3]. Once the neighbors have been found, we can interpolate among the ratings of that neighborhood to predict ratings and make recommendations.

One similarity measure which has commonly been used in recommender systems is Pearson Correlation [15, 12]. It takes into account differing biases in ratings between the users under consideration, based only on items rated in common. Other similarity measures include vector similarity [3]. Our system's nearest neighborhood recommendation scheme can be easily modified to use any such measure.

3 Approach

3.1 Ratings data

In preliminary experiments, we determined that having the user provide feedback to substantiate their rating was effective in improving the mean absolute error of predictions [9]. That preliminary study asked the user to indicate

the presence of a feature and specify a numerical rating regarding their liking or dislike of it. We have observed that the extent to which a feature is applicable is an important criterion for both user satisfaction and performance. In addition, a user's reaction to a feature can depend significantly on the strength of its observed "presence" (i.e. a little bit of violence may be good, but a lot may be negative). Based on these observations we permit a user to:

1. specify an overall opinion of the item,
2. select a relevant feature of the item,
3. specify the quantity of that feature in the item (or applicability of the feature to the item),
4. specify the degree to which the presence of this feature was a positive or negative factor.

This provides a more natural transliteration of the form of a typical interpersonal dialogue regarding the review of an item, but it does impose more overhead on the user. For example,

> *I thought that movie was pretty good. There was a lot of action and special effects, which is great. It's just too bad that the romantic subplot was underdeveloped.*

becomes

Overall: 8

action	quantity 8	opinion 5
romantic subplot	quantity 2	opinion -4

Precisely, a rating by user u on item i is of the following form:

Exactly one *overall* rating, $r_{ui} \in [1, 10]$. This represents the user's opinion of i on the whole, where a rating of 1 indicates extreme dislike and a 10 indicates extreme preference.

For this item, u must select a minimum of one feature which was important to his or her overall opinion, and in general will choose a set of features F_{ui}. Suppose feature $f \in F_{ui}$ is chosen. Then the user specifies the *feature quantity* of f perceived to be in the item, $q^f_{ui} \in [0, 10]$, where 0 indicates the complete absence of this feature while 10 indicates a very large amount. To complete this feature rating, the user must specify his or her *feature opinion* of this presence of f in the item, as $o^f_{ui} \in [-5, 5]$, where a rating of -5 should be used for extremely negative features, and 5 should be used for extremely positive features.

Among users of the system who are not affiliated with the research project, 58% of all ratings used 3 or more features.

The set of all users is U, the set of all items I, and the set of all features in the system, F. For convenience, we denote the set of all items rated by user u as I_u, and the set of all features used by user u as F_u.

3.2 Hybrid CF using feature biases

The system uses a nearest-neighbor interpolation scheme to recommend items that are preferred by similar users. Typically, user similarity is computed as the Pearson correlation [3] of items rated in common between the users. In our system the ratings are more complicated so while we also use Pearson correlation, we actually compute three different similarity measures and then take a weighted average of the three to arrive at one similarity value.

Suppose we wish to compute the similarity between users u and w.

First, we have the similarity in overall ratings, which is just the Pearson correlation between the overall ratings of the items rated in common.

$$s_r(u,w) = \frac{\sum_{i \in I_C}(r_{ui} - \bar{r}_u)(r_{wi} - \bar{r}_w)}{\sqrt{\sum_{i \in I_C}(r_{ui} - \bar{r}_u)^2 \sum_{i \in I_C}(r_{wi} - \bar{r}_w)^2}} \quad (1)$$

where in each summation, $I_C = I_u \cap I_w$, the items common to both user u and w; and $\bar{r}_u$ is the mean overall item rating for user u.

To make use of feature information, we calculate statistics for feature ratings. In effect we compare user biases toward features, rather than comparing feature usage on an item-by-item basis. Thus, the next similarity measure we compute will be the similarity in feature bias between the two users. For each user u, for each feature used, we calculate the mean feature opinion for feature f over all items, $\bar{o}^f_u$, and the mean opinion over all features, $\bar{o}_u$. Then, using the Pearson correlation:

$$s_o(u,w) = \frac{\sum_{f \in F_C}(\bar{o}^f_u - \bar{o}_u)(\bar{o}^f_w - \bar{o}_w)}{\sqrt{\sum_{f \in F_C}(\bar{o}^f_u - \bar{o}_u)^2 \sum_{f \in F_C}(\bar{o}^f_w - \bar{o}_w)^2}} \quad (2)$$

where in each summation, $F_C = F_u \cap F_w$, the features which have been used by both user u and w.

Finally, we wish to get a third measure based on the similarity in bias toward quantity.

$$s_q(u,w) = \frac{\sum_{f \in F_C}(\bar{q}^f_u - \bar{q}_u)(\bar{q}^f_w - \bar{q}_w)}{\sqrt{\sum_{f \in F_C}(\bar{q}^f_u - \bar{q}_u)^2 \sum_{f \in F_C}(\bar{q}^f_w - \bar{q}_w)^2}} \quad (3)$$

where in each summation $F_C = F_u \cap F_w$, the features which have been used by both user u and w; $\bar{q}_u$ is the mean quantity rating by user u over all features and items; and $\bar{q}^f_u$ is the mean quantity rating from user u for feature f, over all items.

We now want to combine these three similarity measures into one. To do this we compute a weighted sum of s_r, s_o, and s_q using weights ω_r, ω_o, and ω_q, respectively. The final similarity between two users u and w then is:

$$s(u,w) = \frac{\omega_r s_r + \omega_o s_o + \omega_q s_q}{\omega_r + \omega_o + \omega_q} \quad (4)$$

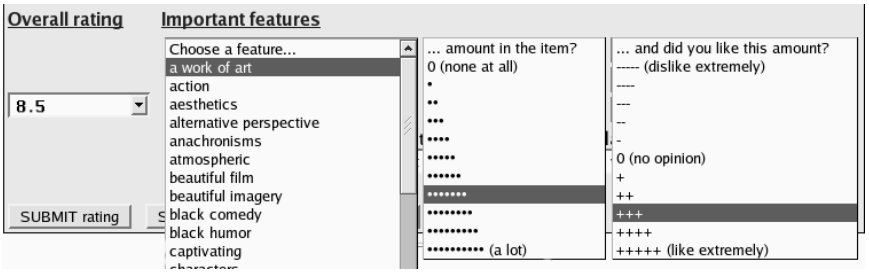

Figure 1. Screenshot of providing feature feedback for a movie in Recommendz.

With the ability to calculate this similarity s between any two users, we can find the k nearest neighbors of user u, which we denote as N_u^k. We will be interested in the subset of N_u^k who have rated item i, denoted N_{ui}^k. With this information, we can predict the rating of user u on item i:

$$\tilde{r}_{ui} = \frac{\sum_{w \in N_{ui}^k} s(u,w)(r_{wi} - \bar{r}_{wi})}{\sum_{w \in N_{ui}^k} s(u,w)} + \bar{r}_u \qquad (5)$$

Note that we normalize the predicted rating according to the users' overall rating biases, and then make the actual prediction by adding to the user's overall mean rating.

3.3. Feature Suggestion

In our system, it is up to the users to contribute features which can then be used by all. Because of this, the number of features in the system is constantly growing. In order to encourage feedback and to be better able to compare user similarity, we would like users to use preexisting features. At the same time we cannot simply present the users with a list of all features in the database, as that would be overwhelming. What we wish to do then is suggest relevant features. The suggested features should be relevant to the item to be rated, and should provide useful information to the system.

Recommendz suggests features using a combination of the following techniques:

Features on which users differ widely in opinion are likely more useful for determining inter-user similarity (this was suggested by Goldberg et al [11]). To exploit this, we suggest features where variance in feature opinion has been high for the current item.

From the features which were not suggested as high-variance features we probabilistically select several more, according to their opinion rating variance. The probability of selecting a feature is directly proportional to the ratio of its variance to the largest variance of all features under consideration.

Based on usage we can calculate the correlation between features. Given these correlations we can suggest features which have not been used on the item in question but which, according to the correlation, are related to features which *have* been used to rate the item. To calculate the correlation between two features f and g, we examine their quantity ratings on items for which both have been used:

$$\text{correlation}(f,g) = \frac{\sum_{i \in I^f \cap I^g} \frac{\bar{q}_i^f \bar{q}_i^g}{(q\text{max})^2}}{|I^f \cap I^g|} \qquad (6)$$

where I^f is the set of all items which feature f has been used to rate, and q_{max} is the maximum quantity rating, in our case 10.

We can also suggest several features which are directly correlated not to features which have actually been used on the item, but which are correlated to those features which are directly correlated to the features used on the item. It is also possible to iteratively suggest features which are correlated to features which in turn are indirectly correlated to the original features of the item.

Suggesting such features allows us to explore the feature space more than focusing entirely on features known to be relevant. We have not yet performed a detailed quantitative analysis to see how well this procedure works, but our initial qualitative observations suggest that it is quite useful.

4 Results

4.1 User response

Since our rating system is new, the question arose as to how users feel about providing feedback. The initial concern was that the detail required might be too complicated, or too time-consuming and arduous. We felt this was a very important issue: The nature of our problem domain is such that accurate predictions will be useless if users are so annoyed by the interface that they never use the system.

In order to determine whether our rating system was in fact a problem, we examined feature usage among users not

affiliated with the research project. We found that the average number of features entered per rated item was roughly 2.5, with just over 58% of items rated with 3 or more features and only 23% of all items rated using the bare minimum of one feature. Some users have used 8, 10, or even more features to rate a single item.

Due to these results, we are confident that *on average* users do not find our system overly complicated or too arduous to use. It is possible that many users find the process of providing detailed feedback fun in and of itself, but such a claim would require collecting user feedback about the system itself, something we have only done informally thus far; however, this hypothesis fits with results reported by Swearingen and Sinha indicating that users of recommender systems are willing to provide more feedback if they feel they are getting something in return [18].

4.2 Experimental results

A commonly used error measure in recommender system research is the normalized mean absolute error (NMAE) (e.g. in [11, 6]). When the maximum and minimum possible overall ratings are $r_{\max}$ and $r_{\min}$, then the normalized mean absolute error for user u is defined as

$$\text{NMAE}(u) = \frac{1}{r_{\max} - r_{\min}} \frac{1}{|I_u|} \sum_{i \in I_u} |\tilde{v}_{ui} - v_{ui}| \qquad (7)$$

To test our system, we used leave-one-out cross-validation over all users who had rated at least 10 items. There were 149 such users at the time of our experiments.

By way of comparison, we examined the global mean rating algorithm ("POP" [3]), in addition to our hybrid CF method described in sections 3.2. The weighting schemes used, along with names by which we will refer to those schemes, are given in table 1. These tests were performed over a range of sizes for the nearest neighborhood. Note that *Pure CF*, *Pure Opinion*, and *Pure Quantity* are not hybrid methods.

Note that in an appendix to [11], Goldberg *et al.* showed that NMAE for prediction by guessing random values was either 0.333 or 0.282, depending on whether the distribution of ratings and predictions were uniform or normal, respectively.

4.2.1 POP prediction

The POP prediction for an item is simply the global mean rating. This is the sort of recommendation to be found in many popular sources (e.g. the metacritic [2] web site).

On our data, the POP algorithm produced an NMAE of 0.234.

[2] http://www.metacritic.com

Table 1. Explanation of the weighting combinations used in testing our approach.

	ω_r	ω_o	ω_q
Pure CF	1	0	0
Pure Opinion	0	1	0
Pure Quantity	0	0	1
Features Only	0	1	1
All	1	1	1
CF+	3	1	1
Opinion+	1	3	1
Quantity+	1	1	3
Features+	1	3	3

4.2.2 Pure CF versus Hybrid CF with feature bias

Pure collaborative filtering (i.e. with no influence given to the feature rating data) outperformed POP predictions, but was in turn outperformed by all of the hybrid schemes except for those which gave the most weight to the feature quantity ratings (*Pure Quantity* and *Quantity+*). See Figure 2 for an illustration of this result.

Among the strictly hybrid methods, illustrated in Fig. 2, we first see that *CF+*, which combines collaborative filtering with a smaller amount of quantity and opinion rating information, improves performance over collaborative filtering alone. As more weight was given to the feature information, especially to feature opinion bias, performance improved further. The results for non-hybrid methods are not illustrated here due to space constraints but are similar, with *Pure Opinion* performing better than traditional collaborative filtering.

5 Conclusions and Discussion

We have described a recommender system which uses descriptive information regarding the items being rated. We show that the use of supplementary descriptive features substantially improves the quality of recommendations over a basic collaborative filter in the domain of movies. For recommendations based on other algorithms, such as sparse factor analysis [6], we expect the same types of improvement can be obtained (preliminary data has been collect to support this conjecture, but it outside the scope of this paper). A key impediment to the use of descriptive features is the need to determine the set to be used and the subset to be presented to a user. We have briefly discussed how this problem can be solved using a combination of algorithmic strategies.

A key aspect of our approach is to obtain supplementary feature information for each item that a user rates. Getting

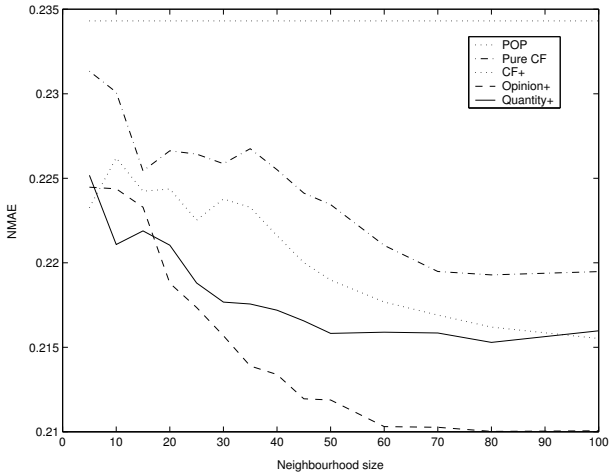

Figure 2. NMAE of various feature bias-based hybrid weighting schemes, as a function of neighborhood size.

such supplementary information flies in the face of the conventional wisdom that users seek to minimize the amount of information they provide. This conventional wisdom is derived, in large measure, from domains in which users do not perceive any direct benefit from the extra effort they must spend. In contrast, our users directly experience enhanced performance from the addition of feature information, both in terms of the quality of the recommendations and presumably from the quality of the personalization they may receive. This is corroborated by the fact that 58 *per cent* of all individual item ratings are accompanied by 3 or more features ratings (despite the fact that the system does not require this much data from a user).

This suggests that users find our feedback system useful. In future work, we would like to quantitatively evaluate the effectiveness of our feature suggestion method, and examine other formulae for computing inter-feature correlation.

While we seem to outperform traditional collaborative filtering schemes, it appears we can do still better by making more use of feature information. In ongoing work, we are examining how to directly exploit feature information to develop algorithmic variations akin to content-based filtering. It also appears that the feature space itself exhibits interesting structure. By making more use of the relationship between features, it seems that we can both improve the quality of the recommendations and provide richer user feedback.

References

[1] C. Boutilier. A POMDP formulation of preference elicitation problems. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pages 239–246, 2002.

[2] C. Boutilier and R. S. Zemel. Online queries for collaborative filtering. In *AI-Stats 2003*. AI and Statistics, January 2003.

[3] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

[4] R. Burke. Semantic ratings and heuristic similarity for collaborative filtering. In *AAAI Workshop on Knowledge-based Electronic Markets*, pages 14–20. AAAI, 2000.

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

[6] J. Canny. Collaborative filtering with privacy via factor analysis. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 238–245. ACM Press, 2002.

[7] S. Dasgupta, W. Lee, and P. Long. A theoretical analysis of query selection for collaborative filtering. *Machine Learning*, 51:283–298, 2003.

[8] J. Dron. CoFIND - an experiment in n-dimensional collaborative filtering. In *Proceedings of WebNet 99*, 1999.

[9] G. Dudek and M. Garden. On user recommendations based on multiple cues. In J. Yao and P. Lingras, editors, *WI/IAT 2003 Workshop on Applications, Products, and Services of Web-based Support Systems*, pages 139–143, October 2003.

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

[11] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.

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

[13] J. Herlocker, J. Konstan, and J. Reidl. Explaining collaborative filtering recommendations. In *ACM 2000 Conference on Computer Supported Cooperative Work*, pages 241–250, December 2000.

[14] M. Montaner, B. López, and J. L. D. L. Rosa. A taxonomy of recommender agents on the internet. *Artificial Intelligence Review*, 19:285–330, June 2003.

[15] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, North Carolina, 1994. ACM.

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

[17] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the tenth international conference on World Wide Web*, pages 285–295, 2001.

[18] K. Swearingen and R. Sinha. Beyond algorithms: An HCI perspective on recommender systems. In *ACM SIGIR 2001 Workshop on Recommender Systems*. ACM SIGIR, 2001.