Mixed Collaborative and Content-Based Filtering with User-Contributed Semantic Features.

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

We describe a recommender system which uses a unique combination of content-based and collaborative methods to suggest items of interest to users, and also to learn and exploit item semantics. Recommender systems typically use techniques from collaborative filtering, in which proximity measures between users are formulated to generate recommendations, or content-based filtering, in which users are compared directly to items. Our approach uses similarity measures between users, but also directly measures the attributes of items that make them appealing to specific users. This can be used to directly make recommendations to users, but equally importantly it allows these recommendations to be justified. We introduce a method for predicting the preference of a user for a movie by estimating the user's attitude toward features with which other users have described that movie.

We show that this method allows for accurate recommendations for a sub-population of users, but not for the entire user population. We describe a hybrid approach in which a user-specific recommendation mechanism is learned and experimentally evaluated. It appears that such a recommender system can achieve significant improvements in accuracy over alternative methods, while also retaining other advantages.

Introduction

In this paper we consider recommendation systems – systems that suggest items of interest to a user – and how semantic information can help them operate. As an experimental domain, we consider a system that suggests movies to subscribers based on information regarding what they have liked and disliked in the past. In general, the standard dogma for such systems is that they should minimize the interaction burden for a user by requiring as little input as possible. Typical examples, including those embedded in internet shopping sites, use only information regarding what movies a user has chosen to see (or what items a user has purchased) to make recommendations. The typical collaborative filtering mechanism matches users to one another if they have similar viewing profiles, and then uses the choices made by one user to generate recommendations for another. Slightly more informed systems use information regarding the degree to which a user may have liked (or disliked) a particular movie. We believe, however, that for optimal accuracy the user may need to provide more comprehensive information.

Clearly, minimizing the burden on users is a laudable objective. All things being equal, we are more likely to prefer a system that imposes limited burdens on us, especially if the rewards for using the system are limited. On the other hand, if our goal is to accurately model a user's preferences and take into account subtle variations in taste and domain, then it is hard to imagine that the limited information described above is sufficient. Clearly, if a psychoanalyst only needed you to rank-order 20 movies to form an accurate model of your psychology, then even Woody Allen would have completed his psychoanalysis by now[1]. In this work we begin with the presumption that users would be willing to engage in a more onerous data collection exercise with an automated system if that system delivered rewards of some kind that made the exercise worthwhile. Indeed, previous work has shown that users are willing to provide more feedback to the system if they feel they are receiving some benefit in return (Swearingen & Sinha 2001).

In this paper, we examine the manner in which supplementary information can be collected from users, used to generate recommendations, and even used to justify the recommendations that are generated. In doing this, an important sub-problem is adaptation to new keywords that may relate to the recommendations to be made. We note also, in passing, that although the example here describes movie recommendations our framework is by-and-large domain independent.

Related Work

Recommender systems

The system called Tapestry is often associated with the genesis of computer-based recommendation and collaborative filtering systems. In Tapestry (Goldberg *et al.* 1992), users were able to annotate documents with arbitrary text comments, and other users could then query based on the comments of other users. The key attribute of this system is that it allowed recommendations to be generated based on a

[1]The writer and director Woody Allen has often recounted the fact that he has been in psychoanalysis for well over 20 years.

synthesis of the input from many other users. Making recommendations based on the opinions of like-minded users rather than filtering items based on content, has become known as *collaborative filtering*. The collaborative filtering paradigm which began with Tapestry was later automated in a number of projects (Resnick & Varian 1997; Balabanovíc & Shoham 1997).

The main advantage of collaborative filtering is the ability to make *serendipitous* recommendations (Herlocker *et al.* 2004). Most systems use a notion of inter-user distance, and thus can define "neighbours" for a user. If an item of a particular genre is highly preferred by a user's neighbour, then that item could be recommended even if the recommendee has no previous experience with items of that genre. For instance, a user who enjoys heavy metal could be recommended a progressive rock album, despite having never heard progressive rock.

One of the primary problems associated with collaborative filtering is that they work best when the number of item rated per user is (unrealistically) high. Since a purely collaborative filter relies on user opinions of items in order to make recommendations, the system will not be able to make recommendations on items for which it has no information (the"cold- start" and "early rater" problems), or for users who are sufficiently dissimilar to all other users.

There are many good examples of collaborative filters and recommender systems. For a survey of current technologies please see (Adomavicius & Tuzhilin 2005).

Content-based filtering

In content-based filtering items are matched either to a user's interest profile or query on the basis of content rather than opinion. One strength of this approach over collaborative filtering is that as long as the system has some information about each item, recommendations can be made even if the system has received a small number of ratings, or none at all. The downside is that each item must be characterized with respect to the features that appear in a user's profile and, further, the profile of each user must be collected and modeled. Naturally, these descriptive features must, themselves, be acquired or engineered somehow.

In work that shares several characteristics with this paper, Claypool et al. (Claypool *et al.* 1999) introduced the p-tango method, which makes use of both a collaborative and content-based filter through a linear combination to produce a more accurate filter than either method alone. In the process of tuning the weighting parameter to find an optimal balance between the two recommendation methods, the authors found that content-based filtering worked better for users with a relatively small number of ratings, for whom it was difficult to accurately estimate a neighbourhood of similar users. Other examinations of hybrid combinations of collaborative and content-based filtering include (Melville, Mooney, & Nagarajan 2002; Burke 2002).

Semantic features

(Burke 2000) suggested that recommender systems might be improved by asking users about desired *features* of items, rather than just for overall ratings of items. In the Entree recommender system users specified a *semantic rating* describing a quality in which a restaurant is lacking, for instance "less expensive", or "quieter". The system uses this feedback to determine what qualities the user is seeking, and to build a profile of each restaurant. It is then possible to recommend restaurants which will fit the user's desires. In Entree, the set of semantic ratings available is determined in advance using manual knowledge engineering.

In the RACOFI music recommender system, multiple preference dimensions are defined in advance, including "originality" and "production" (Anderson *et al.* 2003). The ratings made along these multiple dimensions are used to learn a set of rules which will produce recommendations for a given query.

In both Entree and RACOFI, the semantic features available to be rated by the user are defined in advance by the maintainers of the system. Not only does this limit and constrain the performance of the system, but it may hinder its ability to adapt to new classes of items. Other systems have allowed the user more freedom. In the Recommendz movie recommender system, users contribute all of the semantic features as they use the system (Garden & Dudek 2005). In each rating, the user specifies an overall opinion of the item, and then specifies an arbitrary number of features of the item which were relevant to forming the overall opinion. For each feature, the user rates the "quantity" of the feature in the item, and provides an opinion (i.e. appraisal) regarding this level of the feature in the item. Users can search for items with a high quantity of a particular feature, or receive a list of recommendations made in part using feature rating information. This type of approach forms the basis for the methodology we develop here.

In CoFIND, users can tag items with semantic ratings describing relevant "qualities" (Dron 1999). New semantic ratings can be added to the system by any user, and those which are used frequently are suggested for use. The emphasis in CoFIND is in organizing resources to aid in learning.

There have also been a number of non-academic projects for information organization based on user-contributed semantic features, commonly referred to as "tags". Technorati.com is a search engine for blogs which explicitly makes use of user-contributed semantic features. Flickr.com is an online photograph repository where users can tag photos with arbitrary semantic features, and perform searches against these tags.

Factor analysis The approaches described above ask users to provide information not only about their preferences but also about the content of items, in the form of features. One might ask whether the same feature information could be extracted automatically from the overall ratings using a technique such as Principal Component Analysis (Goldberg *et al.* 2001) or Factor Analysis (Canny 2002). These techniques have proven useful in a variety of fields. One objection to such techniques is that they may lack the explanatory power which has been identified as an important aspect of recommender systems (Herlocker, Konstan, & Reidl 2000; Reilly *et al.* 2005).

Approach

Semantic feature data in Recommendz

In the Recommendz recommender system (described previously in (Garden & Dudek 2005)), users provide feedback on items by rating the item itself, and rating several of the features of the item which the user felt were relevant. Users are free to create arbitrary features in the form of words or short phrases. These features are added to the system and made available for use by other users. Some oversight is maintained by the system administration to merge obvious duplicates (e.g. misspellings).

Formally, the rating of user u on item i, using a set of features F_u^i consists of:

$r_{ui} \in [1, 10]$, the overall rating on the item, with a value of 1 indicating extreme dislike and 10 indicating extreme preference.

For each $f \in F_u^i$:

$q_{ui}^f \in [0, 10]$, the quantity of the feature in the item, or applicability of the feature to the item, with 0 indicating a complete absence and 10 indicating a large presence.

$o_{ui}^f \in [-5, 5]$, the opinion of the user on the presence of f in the item, with -5 indicating an extremely negative effect on the user's overall enjoyment of the item, and a $+5$ indicating an extremely positive effect on the user's overall enjoyment of the item (the use of negative values in this part of the rating is to stress the distinction between features which have a negative versus positive impact, as well as those features which have no importance in preference for the object but are simply being reported as being present with a value of $o_{ui}^f = 0$).

In addition, we denote all features in the system as F, all features used by user u to rate any item as F_u, and all features ever used by any user in rating item i as F^i.

The approach taken in Recommendz is general enough to be used with any item domain. Currently the system offers a database of over 2000 movies, and another database of web-based newspapers, blogs, and podcasts. The experiments reported upon in this paper were made using the Recommendz movie ratings data.

Feature-based filtering

In addition to using the ratings data for collaborative filtering, we wished to explore the use of the same data for content-based filtering. In this approach we estimate user preference toward items directly, rather than by examining the preferences of neighbouring users.

In this approach we build a profile of each user's attitude toward various features and a profile of the features occurring in each item. Given a pair of such profiles we can predict the user's overall rating of the item.

User feature profile We wish to characterize each user in terms of attitudes toward features. We began by presuming that a user's attitude toward a particular feature might vary with the quantity of the feature in question. For instance a user might have a high opinion of slapstick comedy in small amounts but not in large amounts. Keeping this in mind, we summarize a user's attitude to a particular feature as a function of ranges of quantity.

We wish to characterize each user in terms of attitudes toward features by building a model of the expected item rating r_{ui} given the presence of particular features. We assumed that a user's attitude might vary with the quantity in which the feature appears. For this model, feature quantities q_{ui}^f are divided into "low", "medium" and "high" quantity values denoted $\mathtt{q}^{\mathtt{l}}$, $\mathtt{q}^{\mathtt{m}}$, and $\mathtt{q}^{\mathtt{h}}$, respectively. We also experimented with using 1, 5, and 11 bins, and found that the 3 described here performed the best.

"Low", "medium", and "high" are defined in relation to user mean, $\bar{q}_u$, and standard deviation σ_{q_u} for the feature quantity ratings q_{ui}^f of user u. For each feature the user has used more than some minimum number of times, we can then compute three expected values:

$$E[r_{u\cdot}|q_{u\cdot}^f = \mathtt{q}^{\mathtt{l}}] = \text{mean}(r_{ui} \forall \{i : q_{ui}^f < \bar{q}_u - \sigma_{q_u}\})$$
$$E[r_{u\cdot}|q_{u\cdot}^f = \mathtt{q}^{\mathtt{m}}] = \text{mean}(r_{ui} \forall \{i : q_{ui}^f \in \bar{q}_u \pm \sigma_{q_u}\})$$
$$E[r_{u\cdot}|q_{u\cdot}^f = \mathtt{q}^{\mathtt{h}}] = \text{mean}(r_{ui} \forall \{i : q_{ui}^f > \bar{q}_u + \sigma_{q_u}\})$$

The three values represent the expected rating of user u on item i when feature f appears in low, medium, and high quantities, respectively. Originally user feature profiles were computed for each possible quantity value rather than binning by ranges of quantity values, but very poor results were achieved because many profiles were quite sparse.

Item feature profile As described above, for item feature profiles we are interested in expected feature quantities rather than item ratings. We calculate the probability that an arbitrary user will report the feature as appearing in the item with either a low, medium, or high quantity. Given the past ratings for item i and feature f we compute the probabilities that each feature will be reported in "low", "medium", or "high" quantity for the item:

$$Pr(q_{\cdot i}^f = \mathtt{q}^{\mathtt{l}}) = \frac{1}{N_{if}} |\{u : q_{ui}^f < \bar{q} - \sigma_q\}|$$
$$Pr(q_{\cdot i}^f = \mathtt{q}^{\mathtt{m}}) = \frac{1}{N_{if}} |\{u : q_{ui}^f \in \bar{q} \pm \sigma_q\}|$$
$$Pr(q_{\cdot i}^f = \mathtt{q}^{\mathtt{h}}) = \frac{1}{N_{if}} |\{u : q_{ui}^f > \bar{q} + \sigma_q\}|$$

where N_{if} is the total number of times feature f has been used to rate item i, $\bar{q}$ is the mean quantity rating over all quantity ratings in the system, and σ_q is the standard deviation over all quantity ratings in the system. In building this profile we use a global mean and standard deviation since for some users we may have only a few ratings and as a result statistics will not be reliable; in which case we assume similarity to the global population.

User-Item correspondence Given the following information:

Feature profile for user u: $E[r_{u\cdot}|q_{u\cdot}^f = \cdot] \forall f \in F_u$, and

Feature profile for item i: $Pr(q_{\cdot i}^f = \cdot) \forall f \in F^i$

we can calculate

$$E[r_{ui}|f] = \sum_{\mathtt{q} \in Q} E[r_{u\cdot}|q^f_{u\cdot} = \mathtt{q}] Pr(q^f_{\cdot i} = \mathtt{q}) \quad (1)$$

$$\hat{r}_{ui} = \frac{1}{|G|} \sum_{f \in G} E[r_{ui}|f] \quad (2)$$

where $G = F_u \cap F^i$, the set of features used to rate item i and which user u has used, and $Q = \{\mathtt{q^l}, \mathtt{q^m}, \mathtt{q^h}\}$, the possible ranges of quantity values. Each inner summation is the overall rating user u would be expected to give to item i knowing that i has been described with feature f. The value $\hat{r}_{ui}$ is the predicted rating.

Suggesting features

In Recommendz, users of the system contribute all of the semantic features. This makes the rating system extremely flexible, but at the same time the number of features to choose from when providing feedback is very high. In order to minimize this problem the rating interface only presents the user with a subset of features. In order for this approach to be effective, the user u should be presented with features which will actually be useful in rating the item i. In selecting which features to present to a user, there is a tension between several competing requirements.

The system defines a series of functions which generate sets of features to be used as suggestions. It has been suggested (Goldberg *et al.* 2001; Carenini, Smith, & Poole 2003) that features which are highly controversial are good discriminants of user preference; therefore, the function $\mathcal{F}_{H^+}(F^i)$ first selects a set of features which are most controversial in the sense that they have large entropy over the opinion ratings $o^f_{u'i}$ for *all* users u', in ratings of item i. Next, $\mathcal{F}_{H^-}(F^i)$ selects a set of features where the probability of each feature being chosen is proportional to its entropy over opinion ratings.

Next, the function $\mathcal{F}_C(\mathcal{F}_{H^\pm}(F^i))$ selects a set of features which are highly correlated with the highly-controversial features already chosen, but which may not have ever been used to rate item i. The system can continue iterating this process of selecting correlated features. In practice the system repeats this step twice.

$\mathcal{F}_p(F^i)$ selects a subset of the features used most frequently to rate item i. These features are likely to be found useful when rating item i. $\mathcal{F}_p(F_u)$ selects a subset of the features used most frequently by user u. These features are likely to be found usefully by user u when rating any item.

Similarly, $\mathcal{F}_p(F)$ selects a subset of all features in the system which have been used the most frequently, and $\mathcal{F}_H(F)$ selects a subset of the features for which opinion ratings vary the most over all items and users.

Finally, if the number of features returned by all previous functions is below some minimum threshold then the threshold is met by adding features chosen uniformly randomly by the function $\mathcal{F}_R(F)$ (which also allows otherwise ignored features to gain or regain popularity).

The set of features $\mathcal{F}_S(u, i)$ suggested to user u for item i is defined as the union:

$$\mathcal{F}_{H^\pm}(F^i) \cup \mathcal{F}_C(\mathcal{F}_{H^\pm}(F^i)) \cup \mathcal{F}_R(F) \bigcup_{\phi \in \Phi} \mathcal{F}_p(\phi) \quad (3)$$

where $\Phi = \{F^i, F_u, F\}$.

Experimental Evaluation

To determine the effectiveness of this approach, we evaluated the performance of our feature-based recommender in comparison to several other methods, some of which incorporate content information and some of which do not.

Feature-aware collaborative filtering

In (Garden & Dudek 2005), a method was introduced for calculating inter-user similarities through the linear combination of three independent similarity values: the correlation in overall ratings (preference for the same items), in mean feature opinions (preference for the same features), and in mean feature quantities reported (shared tendency to consume items with the same range of quantities for certain features). Users are grouped into neighbourhoods according to this similarity metric and then used to recommend items to other members of the same cluster.

We have duplicated this method as a baseline for comparison and refer to it as `Recommendz` in experimental results. We gave equal weight to overall, opinion, and quantity information, and used a neighbourhood of 75 users. These parameter settings were shown to provide good performance.

Pure collaborative filtering

For comparison with a purely collaborative filtering method we chose the nearest neighbourhood using pearson correlation (Herlocker *et al.* 2004). The maximum neighbourhood size was set to 75 users. This approach is similar to feature-aware collaborative filtering except inter-user similarities are based only on overall ratings. This method is referred to as `CF` in experimental results.

Global mean ("POP")

As a default we used the "POP" recommendation method (Breese, Heckerman, & Kadie 1998), which predicts that each user will rate each item with the global mean rating of all other users. Note that this method also does not consider content information and treats all users as coming from a single neighbourhood. This method is referred to as `POP` in experimental results. We also implemented the rating deviation-based POP method described by (Herlocker *et al.* 2004) but found the accuracy was slightly better, but not significantly so.

For comparison we also provide the results achieved by a user-specific random prediction strategy. In this strategy the prediction for any item rated by user u is drawn randomly from $\mathcal{N}(\bar{r}_u, \sigma_{r_u})$, i.e. a normal distribution sharing the statistics of the user's own overall item ratings.

Error measures

Our performance analysis uses two complementary error measures: the quality of our predicted rank ordering of the items a user has already rated, and the accuracy of our predictions of the specific numerical item ratings. Each of these measures has its merits (Herlocker *et al.* 2004) since users are commonly presented with a ranked list of items, but the numerical ranking is metric data that is actually collected and manipulated.

For accuracy in ranking we used Spearman rank correlation (Herlocker *et al.* 2004), which is just the Pearson correlation computed for the ranks of the ratings, rather than the ratings themselves. Spearman correlation is defined as:

$$\rho = \frac{\sum(x - \bar{x})(y - \bar{y})}{n\sigma_x\sigma_y} \tag{4}$$

where x is the set of actual rankings, y is the set of predicted rankings, and σ_x is the standard deviation of x. It is worth noting, as Herlocker *et al.* point out, that Spearman correlation does not consider that items may have the same rank.

For measuring prediction accuracy we use normalized mean absolute error (NMAE) (Goldberg *et al.* 2001). We use this normalized version of mean absolute error because item ratings in our system are in the range $[1, 10]$ whereas many systems use a range of $[1, 5]$ (see (Herlocker *et al.* 1999) for instance). Normalized mean absolute error is defined as:

$$\text{NMAE} = \frac{\sum_{i=1}^{N} |r_i - \hat{r}_i|}{N(r^\top - r^\perp)} \tag{5}$$

where r_i is the i^{th} of N ratings, $\hat{r}_i$ is the predicted rating for that item, and $r^\top$ and $r^\perp$ are the maximum and minimum ratings, respectively (e.g. in our system $r^\top = 10$ and $r^\perp = 1$).

The *coverage* of a recommender algorithm is the proportion of rated items for which a method could provide a recommendation. We present the coverage of the various methods in the third column of Table 2.

Personalizing the recommender

We found that for a majority of users in each test set, the feature- based recommender consistently produced the most accurate recommendations; however, some users clearly benefitted from the traditional collaborative filtering approach. The percentage of users for which each method was chosen as best is given in Table 1. As a result, we were curious about the extent to which the results could be improved by determining the best recommendation method for each user and then using that method in the future.

Table 1: Percentage of users for which each method produced the best NMAE

Method	
`FB`	60.3%
`Recommendz`	16.7%
`CF`	23.1%

The system's data set was restricted to users with 30 or more ratings, items rated by at least 5 users, and features used by more than one user. This left 885 users, 1641 items, and 667 features, and a total of 19,621 overall ratings and 50,997 feature ratings. A maximum of 50 ratings were randomly selected from each user as test ratings. These ratings were then divided into 5 disjoint subsets. Each of the 5 subsets was successively held back as a set of validation items. The other 4 sets were used to find the best method for the user (in terms of NMAE), and this method was then applied to the held out items to produce the accuracy and coverage results reported here. This method is referred to as `Combined` in experimental results.

The results of our experiments are presented in Table 2. The feature-based method mainly provides an improvement in rating accuracy over pure collaborative filtering, at the cost of loss in coverage. The `Combined` method is more accurate than any other method, especially in terms of its ability to rank-order items correctly.

Coverage was smallest for the feature-based method (see the third column in Table 2). This is to be expected since if a user profile and feature profile have no features in common, then a feature-based prediction cannot be made.

Table 2: Performance summary: NMAE, Spearman rank correlation, and coverage.

Method	NMAE	Spearman ρ	Coverage
`Combined`	0.1840	0.3523	0.88
`FB`	0.1856	0.3235	0.87
`CF`	0.1988	0.3214	0.94
`Recommendz`	0.1995	0.3071	0.94
`POP`	0.2094	0.3078	1.0
`Random`	0.3002	0.0104	1.0

Discussion

We introduced a content-based recommendation method which uses detailed information about user preference for semantic features, and the content of items as reported by users, to predict users' overall opinions on items. We showed that for many users this method provides more accurate results, but a substantial number of users are still better-served by a collaborative filtering approach. By estimating the best method per user even greater accuracy, especially in the ranking task, can be achieved. This suggests that making recommendations to users can and should be personalized in more than one way: both by personalizing the recommendations by taking into account user specific data, but also by personalizing the recommendation algorithm itself.

It remains an open question whether there are semantic, psychological or statistical bases for variation in which recommendation algorithm is best for each user. A preliminary examination of user rating statistics did not reveal any clear correlation between the number of ratings or features used and the choice of algorithm.

Future work

The feature-based rating prediction $\hat{r}_{ui}$ uses a linear combination of estimated ratings; however, it is not clear that a linear combination is necessarily the best method. For instance, it is conceivable that a particular feature, or pair of features, is disproportionately influential, or makes all other features irrelevant. A linear combination of features would not be able to model such a situation. This suggests that other models, for instance techniques from statistical natural language processing, may be applicable here. Applying boosting to the simple $E[r_{u\cdot}|q^f_{u\cdot} = \cdot]$ filters may also provide increased accuracy (Freund & Schapire 1999).

In creating user profiles and item profiles, we currently weight all features equally. Instead, we could use a simple confidence measure based only on the number of ratings used to create the item profile. This confidence measure might be made more accurate by considering the entropy of ratings, in order to boost the influence of features for which there appears to be more agreement among users.

There are reasons to believe that the methods described here have other advantages in terms of both robustness and ease of start-up for new users (since the combined should be able to fall back to feature-based recommendation when insufficient data is available). How these can be exploited and demonstrated objectively remains a subject for future work.

The feature-based recommendation method suffers from a decrease in coverage because of the lack of overlap between user and item profile features. Methods of increasing this coverage level should be investigated. For instance, some features may serve as reasonably accurate synonyms for others. Comparing users and item profiles using both features held in common and features which are correlated could increase coverage without degrading accuracy.

References

Adomavicius, G., and Tuzhilin, A. 2005. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering* 17:734–749.

Anderson, M.; Ball, M.; Boley, H.; Greene, S.; Howse, N.; Lemire, D.; and McGrath, S. 2003. RACOFI: A rule-applying collaborative filtering system. In *Proceedings of COLA'03*. IEEE/WIC.

Balabanovíc, M., and Shoham, Y. 1997. Fab: Content-based, collaborative recommendation. *Communications of the ACM* 40:66–72.

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

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

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

Canny, J. 2002. Collaborative filtering with privacy via factor analysis. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 238–245. ACM Press.

Carenini, G.; Smith, J.; and Poole, D. 2003. Towards more conversational and collaborative recommender systems. In *Proceedings of the 7th International Conference on Intelligent User Interfaces*. IUI.

Claypool, M.; Gokhale, A.; Miranda, T.; Murnikov, P.; Netes, D.; and Sartin, M. 1999. Combining content-based and collaborative filters in an online newspaper. In *ACM SIGIR Workshop on Recommender Systems - Implementation and Evaluation*. ACM SIGIR.

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

Freund, Y., and Schapire, R. 1999. A short introduction to boosting. *Journal of the Japanese Society for Artificial Intelligence* 14:771–780.

Garden, M., and Dudek, G. 2005. Semantic feedback for hybrid recommendations in recommendz. In *Proceedings of the IEEE International Conference on e-Technology, e-Commerce, and e-Service (EEE05)*.

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

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

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

Herlocker, J.; Konstan, J.; Terveen, L.; and Reidl, J. 2004. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems* 22:5–53.

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

Melville, P.; Mooney, R. J.; and Nagarajan, R. 2002. Content-boosted collaborative filtering for improved recommendations. In *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI'02)*, 187–192.

Reilly, J.; McCarthy, K.; McGinty, L.; and Smyth, B. 2005. Explaining compound critiques. *Artificial Intelligence Review* 24:199–220.

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

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