Improving Case-Based Recommendation
A Collaborative Filtering Approach

Derry O' Sullivan, David Wilson, and Barry Smyth

Smart Media Institute
University College Dublin
{dermot.osullivan,david.wilson,barry.smyth}@ucd.ie

Abstract. Data Mining, or Knowledge Discovery as it is also known, is becoming increasingly useful in a wide variety of applications. In the following paper, we look at its use in combating some of the traditional issues faced with recommender systems. We discuss our ongoing work which aims to enhance the performance of PTV, an applied recommender system working in the TV listings domain. This system currently combines the results of separate user-based collaborative and case-based components to recommend programs to users. Our extension to this idea operates on the theory of developing a case-based view of the collaborative component itself. By using data mining techniques to extract relationships between programme items, we can address the sparsity/maintenance problem. We also adopt a unique approach to recommendation ranking which combines user similarities and item similarities to provide more effective recommendation orderings. Experimental results corroborate our ideas, demonstrating the effectiveness of data mining in improving recommender systems by providing similarity knowledge to address sparsity, both at user-based recommendation level and recommendation ranking level.

1 Introduction

Due to the large amount of information present in the world today, technologies are required which filter all the available data, leaving us with only the information which is valuable to us. Such technologies must take into account vast quantities of data as well as scalability issues. Even information sources using application-specific market-basket data suffer this problem; hence recommender systems exist to help us find what we want. Modern recommender systems typically employ collaborative, content-based filtering techniques, or some combination thereof [1,2,3,4,5,6,7]. Content-based filtering techniques operate by analyzing item content and providing recommendations based on this. Genre, for example, could be used as a comparative descriptor for items such as television programs, movies, or music (e.g., *"Friends"* is of genre *comedy* as is *"Frasier"*). Given a partial or complete item description as a query, new items with similar descriptions can be usefully recommended. This provides a means of direct access to related items, as soon as they become available, but it does necessitate a knowledge-engineering overhead in building sufficient item descriptions. Another drawback appears when prioritizing the recommendation of items that are similar to those previously preferred; content-based systems may incur a cost in terms of the diversity of items recommended. In contrast, collaborative filtering(CF) techniques make recommendations by comparing user profiles of rated items, according to similarities and differences in the item ratings. These methods require little knowledge-engineering overhead, and there is greater opportunity for recommendation diversity. However, collaborative techniques incur a cost in gathering enough profile information to make accurate user similarity measurements. This *sparsity problem* tells us that, on average, two users are unlikely to have rated many of the same items and so there will be little direct overlap between their profiles. This is problematic when it comes to retrieval, because it means that there may be no direct way to measure the similarity between two profiles unless we have access to additional knowledge that allows us to compare non-identical profile items. There is also a *latency problem* in that new, one-off, or esoteric items will not be recommended until the items have found their way, if ever, into sufficiently many user profiles.

PTV is an established online recommender system deployed in the television listings domain[4]. PTV already combines complementary results from separate user-based collaborative and case-based components for recommending programs to users. In this research, we are interested in improving recommendation accuracy in the overall system by addressing the sparsity problem in the collaborative component. Sparsity (and latency) can be addressed directly by asking users to make additional rankings tailored to increase recommendation breadth (and depth). However, this presents an unwelcome burden for users, and a given user may have no basis for rating targeted items, which could weaken, rather than strengthen, system recommendations. Thus, we would like to base our efforts on the information provided through the normal task-based processing of the system. For example, we might assign genre information to establish comparative relationships between items beyond item overlap. We are mindful, however, that the collaborative filtering component is desirable, in part, because it can provide diverse and high-quality recommendations with minimal knowledge-engineering effort. One of the goals of this research, then, is to strike a good balance between improving collaborative recommendations and the amount of development overhead in doing so. Our approach applies automatic data mining techniques to extend the similarity relationships between programme items. Addressing the sparsity problem faced by collaborative systems is very much in the spirit of our experiences with reasoning knowledge maintenance in case-based reasoning (CBR) [8]. CBR would typically be viewed well within the content-based recommendation camp, but the methodology of retrieving similar user profiles (cases) and adapting them by combining new items with the current profile (case) context lends itself to a case-based view [9]. With this in mind, we propose a case-based perspective on collaborative filtering [10] which enables CF profiles to be used directly as cases, and we approach the problem of reducing sparsity as a similarity coverage problem in CBR, where similarities are based on an incomplete (possibly non-overlapping) space of descriptors that calls for augmentation (c.f., [8,11]).

This paper describes our data mining approach to improving case-based collaborative recommendation, and it presents experiments that show the benefits of mining similarity knowledge to address sparsity, both at the level of user-based recommendation and at the level of recommendation ranking. From the collaborative perspective, this provides a way to address the sparsity problem. From the CBR perspective, it provides a new approach to similarity maintenance.

2 Mining Programme Similarity Knowledge

There are many automated techniques that could be used to derive item similarity knowledge. The initial approach we have chosen is to apply data mining techniques (see [12] for an overview), in particular the *Apriori* algorithm [13], to extract association rules between programmes in PTV user-profile cases. By discovering hidden relationships between TV programmes, we may be able both to cover more potential profile matches and to make more informed recommendations. For example, under conventional collaborative filtering techniques, a person that likes *X-Files* and *Frasier* would not normally be comparable to a person that likes *Friends* and *ER*, but discovering a relationship between *Frasier* and *Friends* would provide a basis for profile comparison.

Association rules are of the form $A \Rightarrow B$, where A and B are sets of items (television programmes). In data mining terms, whenever a transaction (case) contains a certain itemset (set of programmes) A, then the transaction probably contains another itemset B.

The probability that a given rule holds, *rule confidence*, is the percentage of transactions containing B given that A occurs:

$$P(B \subseteq T | A \subseteq T) \tag{1}$$

The *support* of an itemset A is defined as the fraction of transactions supporting A with respect to the entire database. The support of a rule $A \Rightarrow B$, then, is the probability that both itemsets occur together in a transaction:

$$support(A \Rightarrow B) = P((A \cup B) \subseteq T) \tag{2}$$

The measure of rule confidence is related to support, and can be computed as follows:

$$confidence(A \Rightarrow B) = \frac{support(A \cup B)}{support(A)} \tag{3}$$

In mining association rules, the confidence and support values are often used to constrain exponentially large candidate rule sets by setting thresholds.

2.1 Item Similarities

Treating PTV user profiles as transactions and the rated programmes therein as itemsets, the Apriori algorithm can be used to derive a set of programme-programme association rules and confidence levels. We have limited this initial phase of the work to rules with single-programme antecedents and consequents. Table 1 shows some sample rules that were generated by running Apriori on our PTV data set. The resulting confidence values are taken as probabilities and used to fill in a programme-programme similarity matrix, as shown in Table 2, which provides the additional similarity knowledge necessary to compare non-identical profile items.

Since the matter of additional similarity coverage rests in populating the matrix as densely as possible, two natural extensions suggest themselves. First, the directly generated rules can be chained together ($A \Rightarrow B$ and $B \Rightarrow C$ imply $A \Rightarrow C$) to provide

Table 1. Selected rules for PTV

Rule	Support	Confidence
Friends $\Rightarrow$ Frasier	12%	25%
Friends $\Rightarrow$ ER	14%	37%
Frasier $\Rightarrow$ ER	10%	22%

Table 2. Example item similarity matrix

	Friends	Frasier	ER
Friends	1	.25	.37
Frasier	-	1	.22
ER	-	-	1

indirect programme relationships. A choice then has to be made as to how the indirect rule confidence will be calculated (e.g., minimum, maximum, or some combination of the confidences in the potential paths); our experiments present results from a number of different models. Second, while it is not logically implied, we would like to see whether rule symmetry (e.g., *Friends⇒Frasier* supporting *Frasier⇒Friends*) could be exploited to extend coverage. Based on previous experiments [10], we have decided to ignore this aspect at present, noting it instead for future work. Rule similarity knowledge that is generated by Apriori, we refer to as *direct*, and additional derived knowledge as *indirect*.

Building such item-item similarity knowledge to address the sparsity/maintenance problem is similar in spirit to item-based collaborative techniques [14]. The item-based collaborative approach uses rating overlaps to build item-item similarities, and suggested items are then retrieved in direct comparison to the elements that comprise a user profile. The direct nature of such item retrieval recalls direct content-based item recommendation, as well as the potential cost in terms of diversity.

3 Recommendation Strategy

The availability of item similarity knowledge facilitates a new type of similarity-based recommendation strategy that combines elements from case-based and collaborative recommendation techniques. It facilitates the use of more sophisticated CBR-like similarity metrics on ratings-based profile data, which in turn make it possible to leverage indirect similarities between profile cases, and so generate improved recommendation lists. This new recommendation strategy consists of two basic steps:

1. The target profile, t is compared to each profile case, $s_i \epsilon S$, to select the k most similar cases.
2. The items contained within these selected cases (but absent in the target profile) are ranked according to the relevance to the target, and the r most similar items are returned as recommendations.

3.1 Profile Matching

The profile similarity metric is presented in Equation 4 as the weighted-sum of the similarities between the items in the target and source profile cases. In the situation where there is a direct correspondence between an item in the source, s_i, and the target, t'_j, then maximal similarity is assumed (Equation 5). However, the nature of ratings-based profile cases is such that these direct correspondences are rare and in such situations the similarity value of the source profile item is computed as the mean similarity between this item and the *n* most similar items in the target profile case ($t'_1...t'_n$) (Equation 6).

$$PSim(t, s, n) = \sum_{s_i \epsilon s} w_i \cdot ISim(t, s_i, n) \tag{4}$$

$$ISim(t, s_i, n) = 1 \ if \ t'_j = s_i \tag{5}$$

$$= \frac{\sum_{j=1..n} sim(t'_j, s_i)}{n} \tag{6}$$

Notice, that if $n = 1$ and there is a perfect one-to-one correspondence between the target and source profile cases, then this profile similarity metric is equivalent to the traditional weighted-sum similarity metric used in CBR.

3.2 Recommendation Ranking

Once the k most similar profile cases ($\hat{S}$) to the target have been identified a set of ranked item recommendations can be produced. There are three factors to consider when ranking these recommendations. First, we want to give priority to those items that have a high similarity to the target profile case. Second, items that occur in many of the retrieved profile cases should be preferred to those that occur in few profile cases. Finally, items that are recommended by profiles that are similar to the target should be preferred to items that are recommended by less similar profiles. Accordingly we compute the *relevance* of an item, s_i from a retrieved profile case, *s*, with respect to the target profile, *t*, as shown in Equation 7; where $S' \subseteq \hat{S}$ is the set of retrieved profile cases that contain s_i.

$$Rel(s_i, t, \hat{S}) = ISim(t, s_i, k) \cdot \frac{|S'|}{|\hat{S}|} \cdot \sum_{s \epsilon S'} PSim(t, s, k) \tag{7}$$

4 Experimental Evaluation

In order to evaluate our approach to mining and applying similarity knowledge, we conducted a series of experiments using data from 622 PTV user profiles. The first set of experiments were designed to investigate the performance characteristics of our chosen data mining algorithm within the PTV domain. The second set of experiments tested the potential for mining additional similarity knowledge, in terms of relationships between programme items. The third set of experiments tested the potential of the approach for improving actual recommendation quality. We repeated our experiments using 3 further datasets:

1. MovieLens dataset consisting of 659 user profiles from the movie recommender domain;
2. Fischlar dataset consisting of 650 user profiles similar to PTV (Fischlar [15] is a streaming video library which uses PTV's personalization technologies to provide recommendations).
3. Eachmovie dataset consisting of 651 user profiles similar to MovieLens.

For experimental runs, we examined only positively rated items, leaving negative ratings for future work.

4.1 Tuning Data Mining

In our first set of experiments, we applied the Apriori algorithm to both datasets for different parameterizations of the algorithm. Since data mining was the basis for maintaining similarity knowledge, we wanted to determine how rule generation would be influenced by parameter choice, namely confidence and support thresholds. The first experiment tested the number of rules generated for varying levels of confidence and support. In the PTV data, changes in the number of rules across confidence values for different support levels were quite similar, indicating that the parameterization is more dependent on the measure of confidence than on support. This can be seen more clearly in the results of the second experiment which measured how well the generated rules match the entire profile set. This average accuracy is computed as the ratio of antecedent and consequent matches to antecedent matches. There is little change in rule accuracy as the level of support changes across different levels of confidence. A similar pattern emerged in results from the MovieLens and Fischlar data [10]. Based on these results, we chose representative support levels (PTV and Fischlar: 5%, Eachmovie and MovieLens: 20%) for the remainder of the experiments. Having different representative support values highlights an important difference; both the PTV and Fischlar datasets were much sparser than MovieLens. To corroborate this fact, we use a density metric (Equation 8) taken from [14]:

$$Density(Database) = \frac{Number\ of\ nonzero\ entries}{Number\ of\ total\ entries} \tag{8}$$

where the number of total entries is calculated by multiplying the number of users by the number of items that have been rated at least once; the number of nonzero entries is the total number of ratings overall in the database.

Using this metric, we found PTV is 160% denser than Fischlar, MovieLens 182% denser than Eachmovie and 1100% denser than PTV. This causes lower rule generation at high support values, which necessitates dropping to low support values to find more rules with higher confidences.

4.2 Increasing Similarity Coverage

In the next set of experiments, we were interested in evaluating the degree to which similarity coverage could be improved by the rule sets. For the first experiment, the density of the generated item-item similarity matrix was taken as our measure of similarity coverage. We varied confidence levels from 40% to 5% at 5% intervals. On each run we generated the Apriori direct rule set, as well as a maximal indirect rule set and filled in the item similarity matrix, taking the matrix density relative to the items that participated in the rule set. Results for PTV showed direct item similarities provide an average of 10% coverage, but there is a marked increase for the indirect similarity rules (65% coverage in the best case). Of course, the Apriori method does not generate association rules for every profile programme, and in fact Apriori ignores a great many programmes because their frequency fails the Apriori threshold tests. Nevertheless when we add these newly generated rules to the full item-item matrix, we found an increase in density from 0.6% to 2.6%. Again, a similar pattern was found in the MovieLens (.08% to .36%), Fischlar (.17% to .65%) and Eachmovie (.1% to .35%) datasets [10].

4.3 Improving Recommendations

With encouraging results for increasing the similarity coverage, we designed experiments to test the effect of this new similarity knowledge on recommendation quality. Based on our earlier experiments, we chose representative confidence levels of 10% on the PTV and Fischlar datasets, 60% on Eachmovie and 70% on the MovieLens dataset.

Recommendation Coverage
In the first experiment, we measured the number of profiles in the PTV dataset that could possibly be compared as a percentage of all potential profile comparisons for different similarity metrics. Using our standard collaborative filtering metric gave coverage of 42%, while our similarity metric using direct rules only and then adding indirect rules increased the coverage to 52% and 70% respectively.

Recommendation Accuracy
The final experiment tested the accuracy of recommendation on both datasets. After generating direct and indirect similarity rules, a profile case is selected from the case-base. A parameterized percentage of the items in the selected case are removed from consideration. The remainder of the profile case is then used as the basis for retrieval, using our similarity metric and recommendation ranking. Accuracy was calculated using three metrics:

1. Percentage of removed items that were recommended in each case.
2. Percentage of profiles in which at least one removed item was recommended.
3. A combination of the above: percentage of removed items that were recommended, given a recommendation could be made.

We were most interested in looking for uplift in recommendation quality when comparing our CBR technique (using direct and indirect similarity rules) to a pure collaborative filtering technique. We tested the effect of our recommendation ranking function by correlating rank score/numerical rank and item success. On the PTV dataset, results of the first metric showed that using direct rules (20%) outperformed indirect rules (18%), which in turn outperformed pure CF (8%). A similar pattern was seen in the other metrics

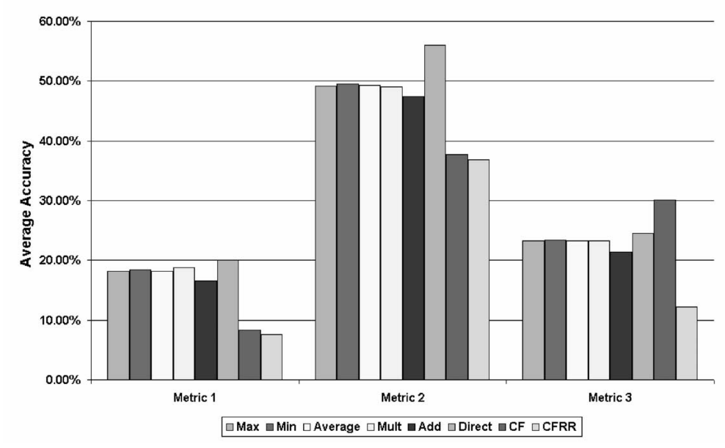

Fig. 1. PTV recommendation accuracy.

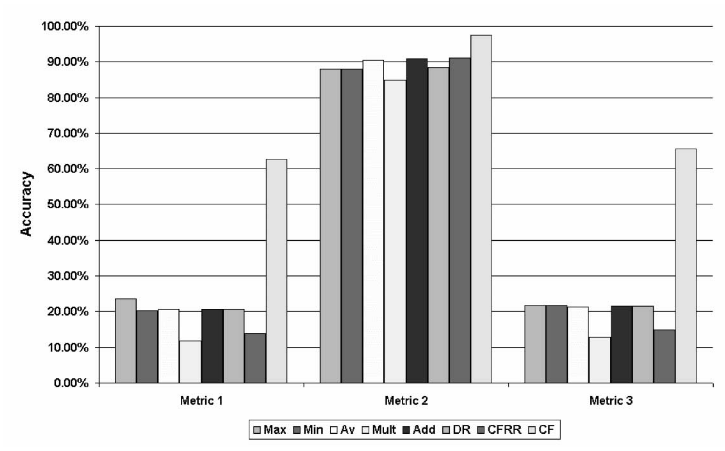

Fig. 2. MovieLens recommendation accuracy.

except for metric 3 which showed pure CF having a higher accuracy (30%) than either CBR method (Direct: 24%, Indirect: 30%) (Figure 1). Fischlar data provided similar evidence to PTV; metric 1 showed rule techniques outperforming their collaborative counterparts (CF: 4%, Direct: 15%, Indirect: 12%). Metrics 2 and 3 confirmed these findings; however metric 3 showed CF (34%) defeating both CBR methods (Direct: 21%, Indirect: 18%) (Figure 3). These results seem to indicate that indirect rules may not be as useful as their direct counterparts; another possibility is that these rules are generating high quality recommendations which are not present in our profile cases.

Eachmovie results showed a similar pattern to the previous two datasets; however, an increase was noted in metric accuracies for all techniques with direct and indirect rules performing better than CF (Figure 4). On the MovieLens dataset, different results

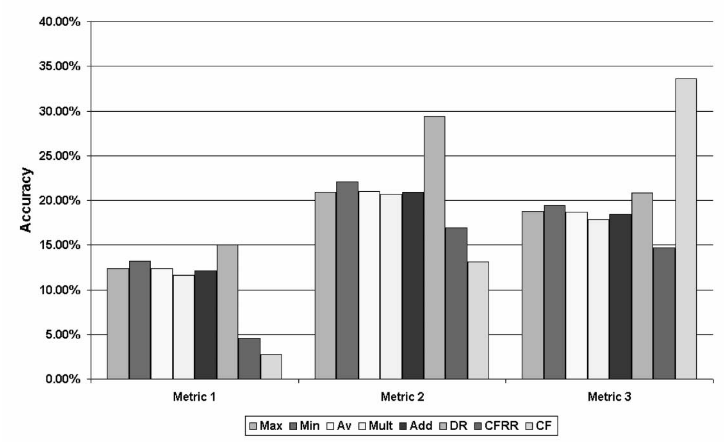

Fig. 3. Fischlar recommendation accuracy.

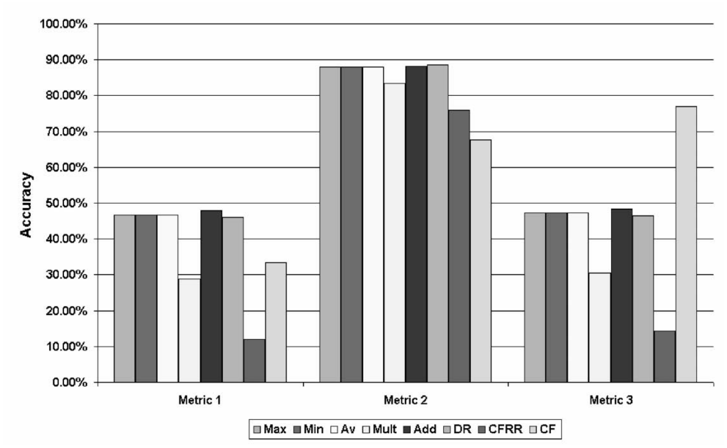

Fig. 4. Eachmovie recommendation accuracy.

were seen. CF outperforms both direct and indirect rules in all three metrics (by almost 30% on average) (Figure 2). These results may indicate that more densely populated profile case-bases favour CF techniques, an issue that bears further study for possible multi-strategy approaches.

Recommendation Ranking Accuracy

In order to test our recommendation ranking function against pure CF methods, we used a simplified version of our recommendation ranking method to test CF ranking: For each unique item in our list of recommended profiles s_i, where $S' \subseteq S$ is the set of retrieved

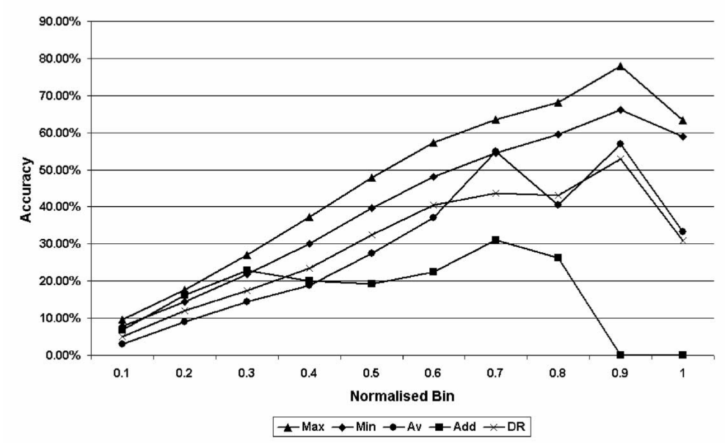

Fig. 5. PTV recommendation rank score accuracy.

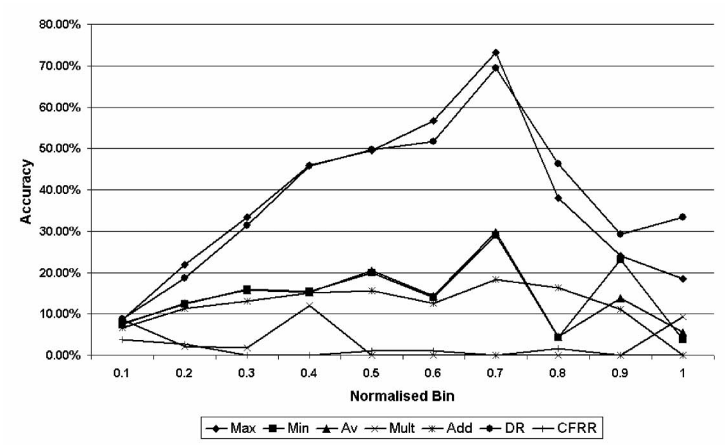

Fig. 6. MovieLens recommendation rank score accuracy.

profiles that contain s_i, and a target t:

$$Rel(s_i, t) = \frac{|S'|}{|S|} \cdot \sum_{s \epsilon S'} CFSim(s, t) \tag{9}$$

We then test our ranking functions by finding correlations between the score/rank of an item and its success over all profiles. For recommendation score, we normalize the score and then find the number of successes in each bin (0 - 1.0 in 0.1 increments); for rank, our bins contain all the successes at each rank position over all profiles. PTV results show that certain models (Maximum, Addition, Average) used in chaining for indirect rule generation give greater correlations over both direct (14% increase rank,

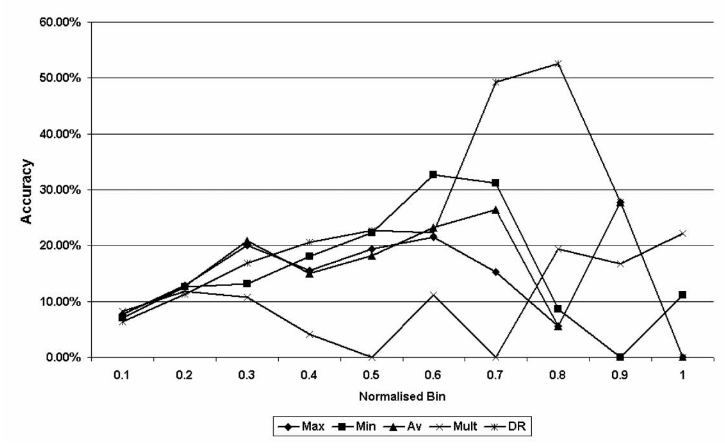

Fig. 7. Fischlar recommendation rank score accuracy.

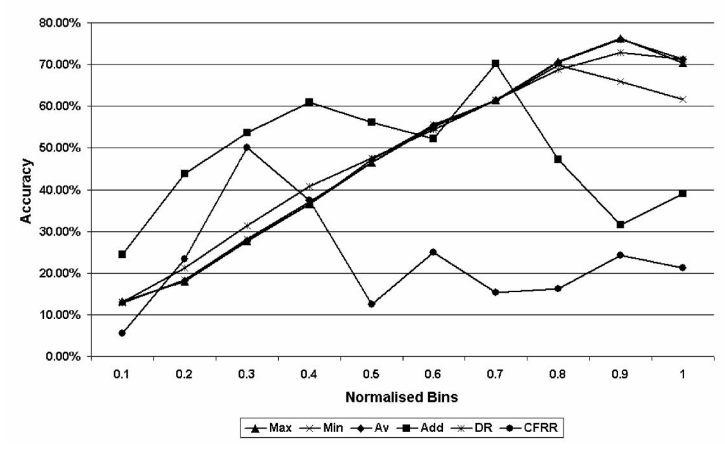

Fig. 8. Eachmovie recommendation rank score accuracy.

19% score) and CF techniques (52% increase rank) (Figure 5). We also found that score binning proved extremely inaccurate under CF with little or no correlation to be found.

With MovieLens data, we again found that certain models provide uplift from direct to indirect rules(8% rank, 4% score) (Figure 6). Again CF results proved inaccurate for use. When comparing both datasets ranking correlations, we also note that using the Maximal model for rule chaining provides the best possible accuracy. Eachmovie results displayed Add, Average and Maximal models performing equally well (Figure 8). With the Fischlar dataset, however, the minimal chaining model gave highest accuracy. Direct rules outperform indirect (14% rank, 24% score) and CF (56% rank) techniques (Figure 7). Future work aims to discover how chaining models affect certain datasets into giving contrasting results.

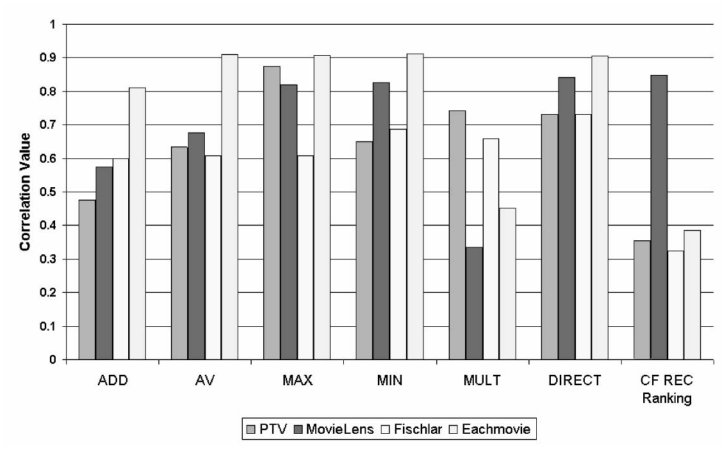

Fig. 9. Recommendation ranking correlations.

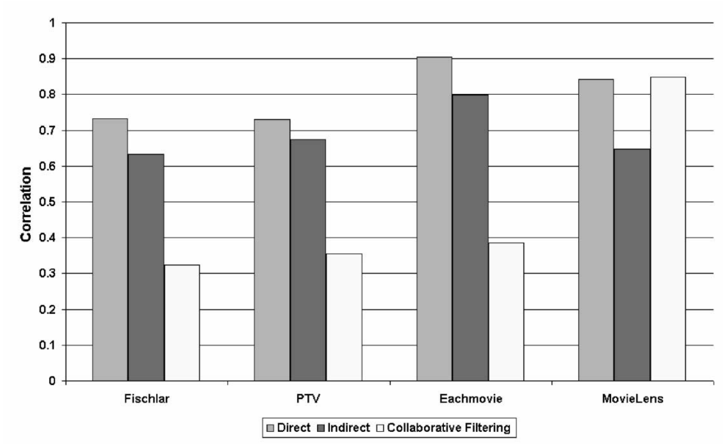

Fig. 10. Accuracy of Techniques.

Figures 10 and 11 confirm a theory discussed previously; the effect of density on techniques used. As the density increases, CF techniques perform comparatively with our techniques; the reverse is true, however, when sparser datasets are used.

5 Conclusion and Future Work

We have described a data mining approach to ameliorating the problem of sparse similarity knowledge in profile-based recommendation systems. From a collaborative filtering standpoint, this provides a means to address the sparsity problem. From a case-based reasoning standpoint, this provides a mechanism for maintaining similarity knowledge. An initial evaluation of the work has been conducted in the domain of TV listing rec-

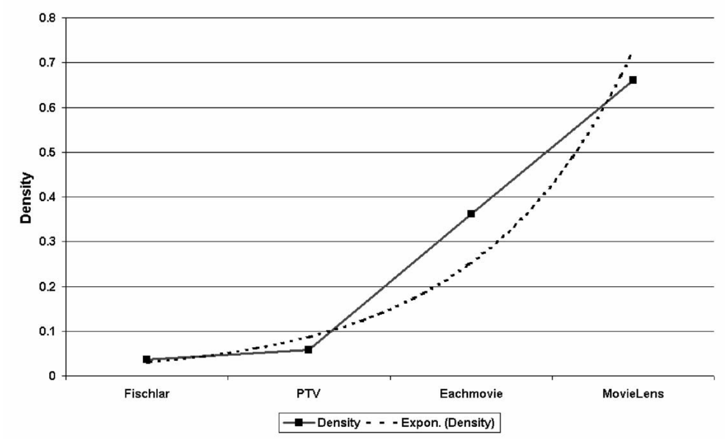

Fig. 11. Dataset Density.

ommendation, using a well known commercial recommender (PTV [4]), and the results are supported by additional experiments, both in the movie recommendation domain and another TV listing recommendation system. The results provide good evidence in support of the view that learning similarity knowledge through profile mining makes it possible to improve overall recommendation quality, especially in sparser datasets.

Our future work will investigate the effects of these algorithms on further data sets to reinforce out theory on density/accuracy, using multiple antecedents and consequents in data mining, inclusion of negative ratings and the use of behavioural data [16] to enhance/test our recommender accuracy. We would also like to compare data mining to item-based collaborative metrics[14], both in measures such as Mean Absolute Error, as well as in terms of recommendation diversity.

References

1. Herlocker, J., Konstan, J., Borchers, A., Riedl, J.: An algorithmic framework for performing collaborative filtering. In: Proceedings of the 1999 Conference on Research and Development in Information Retrieval: SIGIR-99. (1999)
2. Konstan, J., Miller, B., Maltz, D., Herlocker, J., Gordon, L., Riedl, J.: Grouplens: Applying collaborative filtering to usenet news. Communications of the ACM **40** (1997) 77–87
3. Shardanand, U., Maes, P.: Social information filtering: Algorithms for automating "word of mouth". In: Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems. (1995) 210–217
4. Smyth, B., Cotter, P.: Personalized electronic programme guides. Artificial Intelligence Magazine **21** (2001)
5. Good, N., Schafer, J.B., Konstan, J.A., Borchers, A., Sarwar, B.M., Herlocker, J.L., Riedl, J.: Combining collaborative filtering with personal agents for better recommendations. In: Proceedings of the 1999 Conference of the American Association of Artifical Intelligence (AAAI-99). (1999) 439–446

6. Soboroff, I., Nicholas, C.: Combining content and collaboration in text filtering. In: Proceedings of the IJCAI-99 Workshop on Machine Learning for Information Filtering. (1999)
7. Balabanović, M., Shoham, Y.: Fab: Content-based, collaborative recommendation. Communications of the ACM **40** (1997) 66–72
8. Wilson, D.C., Leake, D.B.: Maintaining case-based reasoners: Dimensions and directions. Computational Intelligence **17** (2001)
9. Watson, I.: CBR is a methodology not a technology. Knowledge Based Systems **12** (1999) 303–308
10. O'Sullivan, D., Wilson, D., Smyth, B.: Using collaborative filtering data in case-based recommendation. In: Proceedings of the 15th International FLAIRS Conference. (2002) To Appear.
11. Fox, S., Leake, D.: Learning to refine indexing by introspective reasoning. In: Proceedings of First International Conference on Case-Based Reasoning, Berlin, Springer Verlag (1995) 431–440
12. Hipp, J., Nakhaeizadeh, U.G.: Mining association rules: Deriving a superior algorithm by analyzing today's approaches. In: Proceedings of the 4th European Symposium on Principles of Data Mining and Knowledge Discovery. (2000)
13. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A.I.: Fast discovery of association rules. In Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R., eds.: Advances in Knowledge Discovery and Data Mining. AAAI Press (1996) 307–328
14. Sarwar, B.M., Karypis, G., Konstan, J.A., Riedl, J.: Item-based collaborative filtering recommender algorithms. In: Proceedings of the Tenth International World Wide Web Conference. (2001)
15. Donald, K.M.: (Use of the fischlar video library system)
16. Charu Aggarwal, Zheng Sun, P.S.Y.: Finding profile association rules (1998)