Chapter 9
Capturing Trust in Social Web Applications

John O'Donovan

Abstract The Social Web constitutes a shift in information flow from the traditional Web. Previously, content was provided by the owners of a website, for consumption by the end-user. Nowadays, these websites are being replaced by Social Web applications which are frameworks for the publication of user-provided content. Traditionally, Web content could be 'trusted' to some extent based on the site it originated from. Algorithms such as Google's PageRank were (and still are) used to compute the importance of a website, based on analysis of underlying link topology. In the Social Web, analysis of link topology merely tells us about the importance of the information *framework* which hosts the content. *Consumers* of information still need to know about the importance/reliability of the content they are reading, and therefore about the reliability of the *producers* of that content. Research into trust and reputation of the *producers* of information in the Social Web is still very much in its infancy. Every day, people are forced to make trusting decisions about strangers on the Web based on a very limited amount of information. For example, purchasing a product from an eBay seller with a 'reputation' of 99%, downloading a file from a peer-to-peer application such as Bit-Torrent, or allowing Amazon.com tell you what products you will like. Even something as simple as reading comments on a Web-blog requires the consumer to make a trusting decision about the quality of that information. In all of these example cases, and indeed throughout the Social Web, there is a pressing demand for increased information upon which we can make trusting decisions. This chapter examines the diversity of *sources* from which trust information can be harnessed within Social Web applications and discusses a high level classification of those sources. Three different techniques for harnessing and using trust from a range of sources are presented. These techniques are deployed in two sample Social Web applications—a recommender system and an online auction. In all cases, it is shown that harnessing an increased amount of information upon which to make trust decisions greatly enhances the user experience with the Social Web application.

J. O'Donovan (✉)
Department of Computer Science, University of California, Santa Barbara, California, USA
e-mail: jod@cs.ucsb.edu

J. Golbeck (ed.), *Computing with Social Trust,* Human-Computer Interaction Series,
DOI 10.1007/978-1-84800-356-9_9 © Springer-Verlag London Limited 2009

9.1 Introduction

Long before the World Wide Web, in the 1960s, Harvard professor Stanley Milgram proposed his Small World Experiments [14]. These consisted of a range of experiments designed to test Kochen and Pool's Small World Problem [4]: given a set N of people, what is the probability that each member of N is connected to another member via $k_1, k_2, k_3 ... k_n$ links?

Milgram's test examined the average path length for social networks of people in the United States by using parcel forwarding between source and target individuals via the US postal service. Milgram's experiments revealed that our society is a small world network with shorter-than-expected path lengths, so the probability of two random people knowing each other is less than the expected value. This concept has become more commonly known as the 'six degrees of separation' problem. There have been many protagonists and critics to Milgram's ideas since the 1960s, for example Malcolm Gladwell's popular book 'The Tipping Point' [5] proposes that connectivity in human social networks is largely dependent on a few extraordinary people, whom he calls 'connectors'. (Gladwell's research was based on findings from a range of articles originally published in The New Yorker.)

Of late, Social Web applications such as Wikis and social networking applications like MySpace and Facebook [10] are becoming hugely popular, making experiments such as Milgram's not only more feasible to perform, but also more relevant, since social connectedness can be harnessed and used to enhance the quality of a users experience in the online world.

In 2004 at the outset of my research in this area, a student at Harvard University, Mark Zuckerberg was developing a Social Web application to 'rate peoples looks'. Now in 2008 his application, Facebook, has become a social networking giant with a user base of 67 million. It is currently growing at a rate of 250,000 new users per day[1]. An interesting point of note is that this ubiquitous social networking application, which is bound to have a major impact on social networking research (and society in general), has its roots in the same place that Stanley Milgram performed his small world social networking experiments in the 1960s. A quick search on Facebook reveals many attempts to repeat Milgram's early experiments by getting massive numbers of users to join groups and perform simple tasks. The membership of these groups alone runs into millions of users. To put the growth of social networking into perspective, Milgram posted 296 letters to test his small worlds theory, and only a percentage of these got through to their destination. Facebook has a highly interconnected network of 67 million users who have expressed explicit trust statements about each other. This information has recently been made readily accessible through the Facebook API, which is sure to be a valuable resource for new research on trust within the Social Web.

One of todays foremost thinkers on the topic of network analysis is Alberto Barabási. Barabási hails the analysis of such networks as the 'true science of the

[1] http://www.facebook.com/press/info.php?statistics

future' [3]. In his popular book 'Linked' [2], he describes structural similarities between human social networks, Web topology, hierarchies of species in nature, and a range of other 'scale-free' networks. In 'Linked', the Web is classified as a scale free network because the degree distribution (i.e: the distribution of the number of links to each node) in the topological graph of the Web follows a mathematical 'power law'. A power law is a mathematical distribution usually characterised by a longer tail than a normal or gaussian distribution. Barabasi shows that a wealth of information can be uncovered by analysing the hubs and connectors in Web topology, examining how factors such as growth, node fitness and preferential attachment can define the structure of the Web. The explorations presented in this chapter are partly inspired by Barabasi's ideas but are applied not to the topology of nodes and links of the Web, but the topology of trust relations between the users within another scale free network: the Social Web.

In the Social Web, users are considered the important structure: the reliability of the user providing the information is as important as the information they provide. The notion of 'trust' for users of the Social Web can be viewed analogously to a PageRank in Web search. In a similar manner to each incoming link being considered as a 'vote' in the PageRank algorithm, a trust based algorithm must somehow compute an overall trust score for a user based on some combination of the individual 'units' of trust that the application has as input. Using eBay as an example, a trust algorithm might use the explicit vote given by both parties in a transaction as a unit of trust. The current eBay implementation simply takes the average of these values as the overall trust score. In this chapter we examine ways to gather individual units of trust in the Social Web and examine techniques to combine these values and reintegrate them to benefit users of the Social Web in some way. This work also addresses the issues involved in presentation of trust information to users on the Social Web, from the premise that trust information must be delivered at the right time, with minimal interference for the user.

9.2 Research on Trust in the Social Web

Of late there has been a lot of research attention towards issues of trust and reputation on the Social Web. [19, 23, 9, 1, 12]. This section examines the prominent related research from the perspective of trust modelling within our application domains of online auctions and recommender systems. Looking at the online auction eBay as a first example, Resnick [19] carried out a comprehensive analysis of eBay's trust and reputation system. This survey highlights some interesting findings regarding the nature of trust within the online auction, and the behaviour of its users. Five of Resnick's more salient findings from his work in [19 are listed below (with parenthesised addenda):

1. Despite incentives to free ride, feedback was provided more than half the time. (If people are generally willing to provide feedback most of the time, this means there will be raw data for a trust-modelling system to operate on.)

2. Well beyond reasonable expectation, the feedback was almost always positive. These 'false positives' in raw comment data may effect a trust value between users. The *AuctionRules* algorithm presented later shows how trust values mined from feedback comments can be used to compensate for the unnaturally positive discrete feedback ratings on eBay)
3. Reputation profiles were predictive of future performance. However, the net feedback scores that eBay displays encourages pollyanna assessments of reputations, and is far from the best predictor available.
4. Although sellers with better reputations were more likely to sell their items, they enjoyed no boost in price, at least for the two sets of items that we examined.
5. There was a high correlation between buyer and seller feedback, suggesting that the players reciprocate and retaliate. (It would be interesting to analyse the variance in feedback correlation between the current mechanism and a 'blind' feedback mechanism wherein a user knows the commentee will never see the feedback)

A similar survey to Resnick's was carried out in 2006 by Jøsang et al. in [7]. In this work, Jøsang describes the eBay reputation system as a *collaborative sanctioning system*, which naturally provides incentive for good behaviour of individuals thereby increasing performance of the market as a whole. Other relevant work by Jøsang et al. focuses on the transitive property of trust in online applications in [9, 8]. In their approach, a diverse set of dimensions was constructed to represent trust, and a mathematical notation was defined to manipulate trust values based on this simplification. The dimensions used in [8] are *trust origin*, *trust target* and *trust purpose*. Represented according to these dimensions, Jøsang et al. find that trust does have some properties of transitivity, based on their study of some basic trust topologies (Fig. 9.1).

Figure 9.1 shows two of the core ideas from Jøsang's 2003 work. Part (*a*.) depicts Jøsang's notion of parallel trust combination. In this figure, Alice can assess trust for a third party (David) by making decisions based on input from multiple sources (Bob and Claire). In this example, if Bob and Claire's recommended trust for David

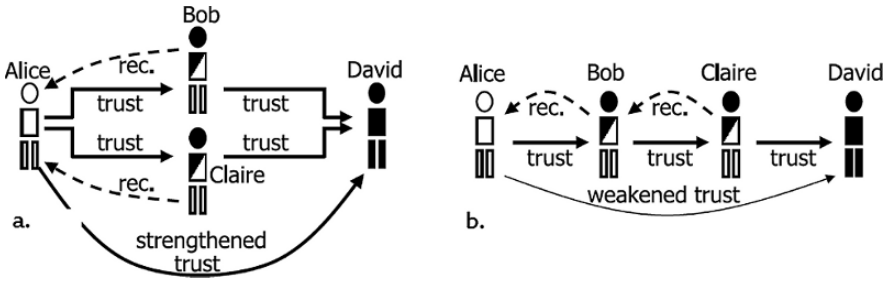

Fig. 9.1 Trust transitivity diagrams from Andun Jøsang's work in [8]. Part (**a**) shows recommendation from multiple sources and resultant parallel trust propagation. Part (**b**) shows basic recommendation and trust propagation across a chain of users

was highly correlated, it would stand to reinforce the derived trust that Alice has for David. Conversely, if Bob and Claire's recommendations about David were highly uncorrelated, it should lower the confidence level placed in the derived trust that Alice has for David. Jøsang deals in detail with various approaches to combine uncorrelated trust expressions of this form. An interesting perspective that was not discussed in [8] is that when Bob and Claire have uncorrelated trust statements for Alice about David, this tells us valuable information about Bob and Claire as recommenders. For example, if it manifests that David is highly trustworthy, and Bob has cautioned Alice that he is not trustworthy, then this misinformation should be remembered the next time Alice receives a recommendation from Bob. Essentially, the mistake should decrease Bobs overall trustworthiness as a recommender.

Figure 9.1 part (*b*.) shows the most basic form of trust transitivity defined in [8]. In this example, 'recommendations' are propagated from Claire, (who directly trusts the target David) through Bob to the source, Alice. As in a real world social or business situation, this results in a weaker trust (less confidence) than if Alice interacted with David directly. Note: In this example the term 'recommendations' is used to represent statements of the form 'I recommend that you should trust this person'.

Research in [22, 23] by Ziegler details the implementation of a propagation model for trust among users in the Social Web. Ziegler focused largely on use of semantic knowledge to achieve trust metrics. A classification scheme for trust metrics is developed in [22], and the benefits and drawbacks of this system are analysed in the context of several Semantic Web scenarios. Ziegler introduces the *Appleseed* system in [22] which is enabled to compute trust for local groups by borrowing ideas from spreading activation models in neuropsychology. According to Ziegler, trust metrics can be broadly categorised into *local* and *global*, where global metrics consider all peers/users and all the links between them. Global trust ranks are computed on the full trust graph, in a manner analogous to the Google PageRank algorithm. Local trust metrics on the other hand, only use partial trust graphs, and constitute *personalised* trust, based on the rational that the set of persons trusted highly by agent *a* may be different from those trusted by agent *b*.

9.3 Trust Sources on the Social Web

Figure 9.2 depicts the application area of this work within the context of the larger World Wide Web. Figure 9.2 also shows some of the application areas which rely heavily on the opinions and views of communities of users and some of the overlaps and interaction areas between these applications. In essence, we are saying that any application on the Social Web, which relies on the opinions, contributions or actions of communities of users, stands to benefit from analysis of the underlying trust relationships that exist in that community.

Trust is a difficult concept to define, and there are many, sometimes conflicting definitions from a range of disciplines. For the work presented in this chapter, we

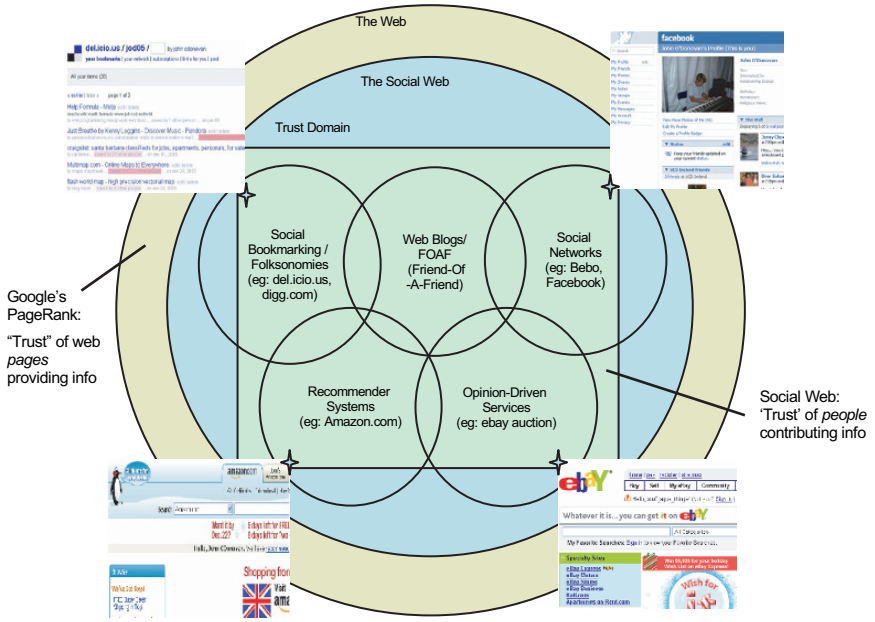

Fig. 9.2 Trust domain within the social Web

are mainly interested in trust in a *computational* sense, as explained by Marsh in [11], particularly, that trust can be viewed as a function of reputation, which can be computed over historical data.

Figure 9.3 provides an illustration of the general concept used in each of the three trust-based applications presented: Users interact with the Social Web application and provide information such as item ratings, product purchases, feedback comments about other users etc. Data-mining techniques can be applied to this data to compute models of trust between individual users. Information from these models can then be re-integrated into the existing mechanisms of the Social Web application to improve the user experience in some way. A high level classification of trust sources within Social Web applications is presented in Table 9.1. Two distinct types of trust are highlighted: *indirect-trust* is a value or set of values to represent trust, which has been computed using some proxy. Examples of such proxies include ratings, profile info, purchases and so on. *Direct-trust* is a value, opinion or comment expressed by one user directly about another during their interaction with the system, for example eBay feedback comments or facebook 'friends'. Research by Jøsang [7] discusses direct and indirect trust in the context of trust propagation between users on the Social Web, this is briefly discussed in the related work section. Jøsang's discussion focuses on the transitive properties of trust as it is passed from person to person. A distinction is drawn between Jøsang's concept and that illustrated in Fig. 9.3 where *indirect trust* uses items as a proxy for trust information between users. For instance, a rating of a movie or a purchased product.

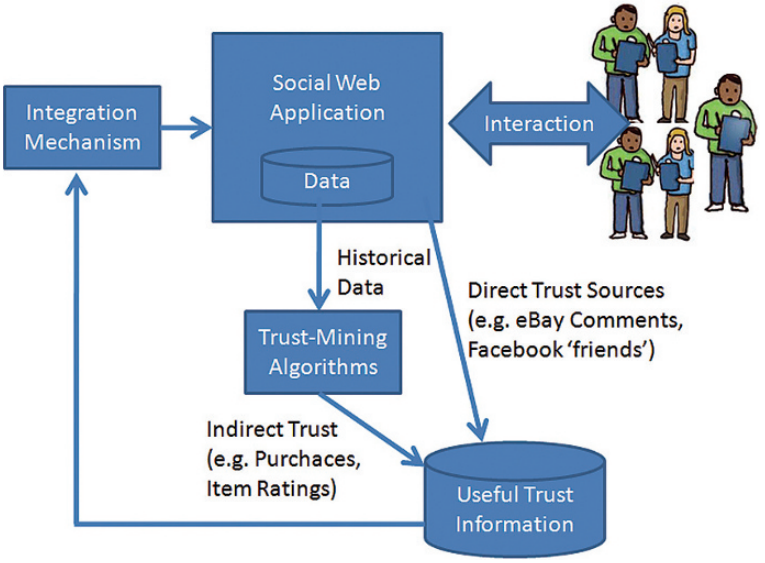

Fig. 9.3 Sources of trust in social Web applications

Table 9.1 Overview of three trust-based algorithms with application domain and classification of trust sources shown for each algorithm

Method	algorithm name	domain	Trust type	original input
1	tRec	ACF Systems	Indirect	Inattentive
2	AuctionRules	Feedback Systems, eg eBay	Direct	Attentive
3	PeerChooser	(Visual) ACF Systems	Both	Both

Table 9.1 expresses a high-level classification of trust sources within Social Web applications, and provides an overview of each of the three the techniques presented in this chapter, showing for each technique the trust type, source and domain. Source type indicates whether or not the initial indication of trust was *attentive* (e.g.: leaving a positive comment on eBay about a particular transaction partner) or implicit (e.g.: purchasing a product, following links etc.). For the purpose of our classification of trust sources, a source in which a user is not aware they are providing trust information is termed an *inattentive* trust source. In Table 9.1, trust type can be either *direct* or *indirect* and indicates whether the trust was expressed by one user directly about another, or computed using some proxy, as discussed above.

The first technique shown in Table 9.1 is a trust modelling algorithm for a collaborative (ACF) recommender system. In this technique, trust is computed over a historical dataset of ratings on movies. For the purpose of this discussion, we refer to these movie ratings as implicit- although the user has explicitly rated a movie, they have no knowledge that this information can be used to compute a trust value about them. This is also an indirect source of trust since the user is not directly making a trust statement about another user. The second technique

presented takes an entirely different approach to harnessing trust information. The *AuctionRules* algorithm computes trust values between users by mining sentiment information from freetext comments on an online auction site. These types of trust can be considered as direct, since the original trust source is essentially a satisfaction statement made by one user directly about another. The source of trust in *AuctionRules* is considered as *attentive*, since although the comments are processed by an NLP algorithm, a user has made an knowing effort to enter the original feedback comment and express a view about another user. The final trust source presented in this work can be viewed a combination of sources. *PeerChooser* is an interactive interface for an ACF recommender system in which users can view a personalised set of their neighbours trust values (computed by technique 1), and modify their values at recommendation time. This can help a user to express current mood and requirements for example. The trust sources used by the *PeerChooser* interface can be classified as a combination of both direct and indirect, since they are originally computed using item ratings, but then modified directly by the user in an interactive interface.

The remainder of this chapter presents the approach and evaluation of the three techniques used for sourcing trust in two Social Web applications, recommender systems and online auctions. This is followed by a comparative analysis of each technique with respect to the source of the trust information used.

9.4 Source 1: Modelling Trust from Ratings in ACF Recommender Systems

Recommender systems are a tool for tailoring large quantities of information to suit the needs of individual users. These systems are widely used on the Social Web, for instance Amazon.com recommends books to millions of users every day. In this section a technique is discussed for modelling trust from user ratings within an Automated Collaborative Filtering (ACF) recommender system. To date ACF systems have relied heavily on what might be termed the *similarity assumption*: that similar profiles (similar in terms of their ratings histories) make good recommendation partners. However, in the real world, a person seeking a recommendation would look further than just similarity with a potential recommender. For example, my friend Bob is a mechanic and is quite similar to me in many respects, but I happen to know from past experience that he has a limited knowledge of computer science, and because of this I would never seek a recommendation from him on anything related to computer science. However, I would trust him highly to recommend anything related to cars, as he has proven to be highly competent in this area in the past. We believe that similarity alone is not enough to base recommendations on in the online world. Trust is an important factor in deciding who to receive recommendations from in Social Web systems as well as in real world situations. Following is a description of the design, implementation and evaluation of a novel technique for modelling trust in ACF recommender systems.

A distinction is drawn between two types of profiles in the context of a given recommendation session or rating prediction. The *consumer* refers to the profile receiving the item rating, whereas the *producer* refers to the profile that has been selected as a recommendation partner for the consumer and that is participating in the recommendation session. So, to generate a predicted rating for item i for some consumer c, we will typically draw on the services of a number of producer profiles, combining their individual recommendations according to some suitable function, such as Resnick's formula, for example (see Eq. 9.1).

Our benchmark algorithm uses Resnick's standard prediction formula which operates by computing a weighted average of deviations from each neighbour's mean rating, and which is reproduced below as Eq. 9.1; see also [18]. In this formula $c(i)$ is the rating to be predicted for item i in consumer profile c and $p(i)$ is the rating for item i by a producer profile p who has rated i, $P(i)$. In addition, $\overline{c}$ and $\overline{p}$ refers to the mean ratings for c and p respectively. The weighting factor $sim(c, p)$ is a measure of the *similarity* between profiles c and p, which is traditionally calculated as Pearson's correlation coefficient.

$$c(i) = \overline{c} + \frac{\sum_{p \epsilon P(i)} (p(i) - \overline{p}) sim(c, p)}{\sum_{p \epsilon P_i} |sim(c, p)|} \tag{9.1}$$

Item predictions are generated using benchmark correlation and prediction methods as it allows for ease of comparison with existing systems. As we have seen above Resnick's prediction formula discounts the contribution of a partner's prediction according to its degree of similarity with the target user so that more similar partners have a larger impact on the final ratings prediction.

The key to our technique is an assessment of the reliability of such partners profiles to deliver accurate recommendations in the future. This is achieved by assuming continuity and assessing the quality of each partner's contribution to recommendations in the past. Intuitively, if a profile has made lots of accurate predictions in the past, they can be viewed as more trustworthy that another profile that has made many poor predictions. In this section we define two models of trust and show how they can be readily incorporated into the mechanics of a standard collaborative filtering recommender system.

9.4.1 Combining Trust in ACF

Figure 9.4 shows a graphical overview of the trust-based recommendation process. In this figure, trust is integrated into a standard ACF algorithm. Initially, trust and similarity are calculated for every pair of users in the database in an offline process. There are a range of ways in which trust is computed, resulting in different models. These are discussed in detail in Section 9.4.2, for now the focus is on integration

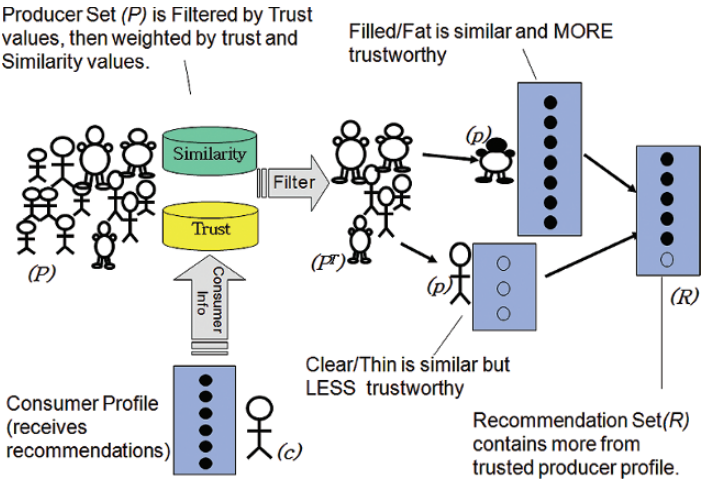

Fig. 9.4 Integrating trust models into standard collaborative filtering

of trust into ACF. During the recommendation process, peer groups are firstly constructed based on a similarity function and then they are filtered based on trust. In Fig. 9.4, C is the active user (or 'consumer') who is seeking recommendations from the system. P is the set of potential recommendation partners (also known as the 'neighbourhood', 'peergroup' or 'recommendation producers') who are most highly correlated and therefore 'similar' to the active user. A personalised trust score is retrieved from a database for the consumer and each potential recommendation partner in turn. If this value is below a certain threshold, that potential producer is dropped from the candidate set. In this manner, only highly similar and highly trustworthy users from the final set of recommendation producers P_t.

Now that the set of producer profiles has been selected, the problem of combining their individual contributions must be addressed. To give a quick example, if I had two highly trusted friends who both were computer scientists, and I needed a recommendation on the quality of a new programming language, I would listen to both opinions, but most likely I would give more weight to the recommendation from person who had the better reputation in the field of programming languages. Accordingly, in Fig. 9.4 the contribution of each member of the trusted producer set P_t is weighted according to the associated trust value. In this manner, more trusted users (indicated as the fatter users in Fig. 9.4) have more influence over the final recommendation set R that is presented to the consumer. Section 9.4.3 provides a detailed description of the specific algorithms which have been developed to combine trust with similarity in the recommendation process.

9.4.2 Capturing Profile-Level & Item-Level Trust

Figure 9.5 shows a high-level overview of the computation described in this section. A ratings prediction for an item, i, by a producer p for a consumer c, is deemed

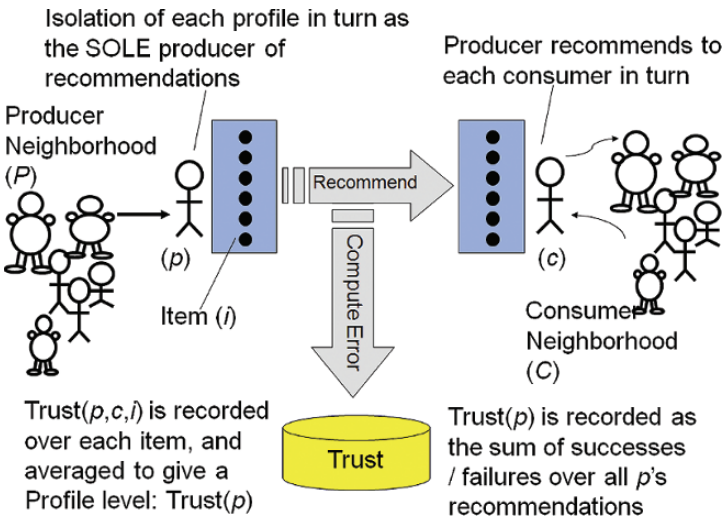

Fig. 9.5 Overview of the trust building process

correct if the predicted rating, $p(i)$, is within ϵ of c's actual rating $c(i)$; see Eq. 9.2. Of course normally when a producer is involved in the recommendation process they are participating with a number of other recommendation partners and it may not be possible to judge wether the final recommendation is correct as a result of p's contribution. Accordingly, when calculating the correctness of p's recommendation we separately perform the recommendation process by using p as c's sole recommendation partner. For example, a trust score for item *i1* is generated for producer *b* by using the information in profile *b only* to generate predictions for each consumer profile. Equation 9.3 shows how a binary success/fail score is determined depending on whether or not the generated rating is within a distance of ϵ from the actual rating a particular consumer has for that item. In a real-time recommender system, trust values for producers could be easily created on the fly, by a comparison between our predicted rating (based only on one producer profile) and the actual rating which a user enters.

$$Correct(i, p, c) \Leftrightarrow |p(i) - c(i)| < \epsilon \tag{9.2}$$

$$T_p(i, c) = Correct(i, p, c) \tag{9.3}$$

From this we can define two basic trust metrics based on the relative number of correct recommendations that a given producer has made. The full set of recommendations that a given producer has been involved in, $RecSet(p)$, is given by Equation 9.4. And the subset of these that are correct, $CorrSet(p)$ is given by Equation 9.5.

$$RecSet(p) = \{(c_1, i_1), ..., (c_n, i_n)\} \tag{9.4}$$

$$CorrSet(p) = \{(c_k, i_k) \in RecSet(p) : Correct(i_k, p, c_k)\} \tag{9.5}$$

The *profile-level trust*, $Trust^P$ for a producer is the percentage of correct recommendations that this producer has contributed; see Equation 9.5. Obviously, profile-level trust is very coarse grained measure of trust as it applies to the profile as a whole.

$$Trust^P(p) = \frac{|CorrSet(p)|}{|RecSet(p)|} \tag{9.6}$$

Suppose a producer has been involved in 100 recommendations, that is they have served as a recommendation partner 100 times, and for 40 of these recommendations the producer was capable of predicting a correct rating, the profile level trust score for this user is 0.4. In reality, we might expect that a given producer profile may be more trustworthy when it comes to predicting ratings for certain items than for others. Accordingly we can define a more fine-grained item-level trust metric, $Trust^I$, as shown in Equation 9.7, which measures the percentage of recommendations for an item i that were correct.

$$Trust^I(p, i) = \frac{|\{(c_k, i_k) \in CorrSet(p) : i_k = i\}|}{|\{(c_k, i_k) \in RecSet(p) : i_k = i\}|} \tag{9.7}$$

9.4.3 Trust-Based Recommendation

Section 9.4.1 presented a high level graphical overview of the process by which trust is integrated into an ACF algorithm. Now we present details of three specific techniques for harnessing trust to achieve better recommendations. We will consider 2 distinct adaptations of a standard ACF technique: *trust-based weighting* and *trust-based filtering*, both of which can be used with either profile-level or item-level trust metrics. The third technique presented is a hybrid of the first two adaptations.

9.4.3.1 Trust-Based Weighting

Perhaps the simplest way to incorporate trust in to the recommendation process is to combine trust and similarity to produce a compound weighing that can be used by Resnick's formula; see Eq. 9.8.

$$c(i) = \overline{c} + \frac{\sum_{p \epsilon P(i)} (p(i) - \overline{p}) w(c, p, i)}{\sum_{p \epsilon P(i)} |w(c, p, i)|} \tag{9.8}$$

$$w(c, p, i) = \frac{2(sim(c, p))(trust^I(p, i))}{sim(c, p) + trust^I(p, i)} \tag{9.9}$$

For example, when predicting the rating for item i for consumer c we could compute the arithmetic mean of the trust value (profile-level or item-level) and the similarity value for each producer profile. A modification on this has been made by using the harmonic mean of trust and similarity; see Eq. 9.9 which combines profile similarity with item-level trust in this case. The advantage of using the harmonic mean is that it is robust to large differences between the inputs so that a high weighting will only be produced if both trust and similarity scores are high. Algorithm 1 is a pseudocode representation of the prediction process using trust-based weighing.

Input: Set of producer profiles P, trust model T, consumer profile c
Output: item recommendation for consumer profile
foreach *Producer Profile p* **do**
  foreach *CandidateItem i* **do**
    w(c,p,i) = harmonicMean(sim(c,p), trust(p,i)) ;
    sum = sum + (p(i)-p)w(c,p,i) ;
    div = div + abs(sim(c,p)) ;
  end
  RecList(i) = ConsumersAverageRating + (sum/div) ;
end
return sortByValue(RecList)

Algorithm 1: Prediction using trust-based weighting

9.4.3.2 Trust-Based Filtering

As an alternative to the trust-based weighting scheme above we can use trust as a means of filtering profiles prior to recommendation so that only the most trustworthy profiles participate in the prediction process. For example, Eq. 9.10 shows a modified version of Resnick's formula which only allows producer profiles to participate in the recommendation process if their trust values exceed some predefined threshold; see Eq. 9.11 which uses item-level trust ($Trust^I(p, i)$) but can be easily adapted to use profile-level trust. The standard Resnick method is thus only applied to the most trustworthy profiles. Pseudocode for the prediction process using trust-based filtering is shown in 9.10.

$$c(i) = \overline{c} + \frac{\sum_{p \epsilon P^T(i)} (p(i) - \overline{p}) sim(c, p)}{\sum_{p \epsilon P^T(i)} |sim(c, p)|} \tag{9.10}$$

$$P_i^T = \{p \epsilon P(i) : Trust^I(p, i) > T\} \tag{9.11}$$

```
Input: Set of producer profiles P, trust model T, consumer profile c
Output: item recommendation for consumer profile
foreach Producer Profile p do
    if Trust(i,p,c) ≥ threshold then
        foreach CandidateItem i do
            sum = sum + (p(i)-pAvg)sim(c,p) ;
            div = div + abs(sim(c,p)) ;
        end
        RecList[i] = ConsumersAverageRating + (sum/div) ;
    end
end
return sortByValue(RecList) ;
```

Algorithm 2: Prediction using trust-based filtering

9.4.3.3 Combining Trust-Based Weighting and Filtering

Of course, it is obviously straightforward to combine both of these schemes so that profiles are first filtered according to their trust values and the trust values of these highly trustworthy profiles are combined with profile similarity during prediction. For instance, Eq. 9.12 shows both approaches used in combination using item-level trust.

$$c(i) = \overline{c} + \frac{\sum_{p \epsilon P^T(i)} (p(i) - \overline{p}) w(c, p, i)}{\sum_{p \epsilon P^T(i)} |w(c, p, i)|} \tag{9.12}$$

9.4.4 Evaluation

So far the argument has been that profile similarity alone may not be enough to guarantee high quality predictions and recommendations in collaborative filtering systems. Trust has been highlighted as an additional factor to consider in weighting the relative contributions of profiles during ratings prediction. In the discussion section all of the important practical benefits of incorporating models of trust into the recommendation process are analysed. Specifically, a set of experiments conducted to better understand how trust might improve recommendation accuracy and prediction error relative to more traditional collaborative filtering approaches is described.

This experiment uses the standard MovieLens dataset [18]. This set contains 943 profiles of movie ratings. Profile sizes vary from 18 to 706 with an average size of 105. We divide these profiles into two groups: 80% are used as the producer profiles and the remaining 20% are used as the consumer (test) profiles.

Before evaluating the accuracy of the new trust-based prediction techniques, trust values must firstly be build up for the producer profiles as described in the next section. It is worth noting that ordinarily these trust values would be built on-the-fly during the normal operation of the recommender system, but for the purpose of this experiment they have been constructed separately in an off-line process, but without reference to the test profiles. Having built the trust values, the effectiveness of our new techniques are evaluated by generating rating predictions for each item in each consumer profile by using the producer profiles as recommendation partners. This is done by using the following different recommendation strategies:

1. *Std* - The standard Resnick prediction method.
2. *WProfile* - Trust-based weighting using profile-level trust.
3. *WItem* - Trust-based weighting using item-level trust.
4. *FProfile* - Trust-based filtering using profile-level trust and with the mean profile-level trust across the producers used as a threshold..
5. *FItem* - Trust-based filtering using item-level trust and with the mean item-level trust value across the profiles used as a threshold.
6. *CProfile* - Combined trust-based filtering & weighting using profile-level trust.
7. *CItem* - Combined trust-based filtering & weighting using item-level trust.

9.4.5 Building Trust

Ordinarily the proposed trust-based recommendation strategies contemplate the calculation of relevant trust values on-the-fly as part of the normal recommendation process or during the training phase for new users. However, for the purpose of this study, trust values must be calculated in advance. This is done by running a standard *leave-one-out* training session over the producer profiles. In short, each producer temporarily serves as a consumer profile and rating predictions are generated for each of its items by using Resnick's prediction formula with each remaining producer as a lone recommendation partner; that is, each producer is used in isolation to make a prediction. By comparing the predicted rating to the known actual rating we can determine whether or not a given producer has made a correct recommendation – in the sense that the predicted rating is within a set threshold of the actual rating—and so build up the profile-level and item-level trust scores across the producer profiles.

This approach is used to build both profile-level trust values and item-level trust values. To get a sense of the type of trust values generated histograms of the profile-level and item-level values for the producer profiles in Fig. 9.6 and 9.7 are presented. In each case we find that the trust values are normally distributed but they differ in the degree of variation that is evident. Not surprisingly there is greater variability in the more numerous item-level trust values, which extend from as low as 0.5 to as high as 1. This variation is lost in the averaging process that is used to build the profile-level trust values from these item-level data. Most of the profile-level trust values range from about 0.3 to about 0.8. For example, in Fig. 9.7 approximately 13% of profiles have trust values less that 0.4 and 25% of profiles have trust values

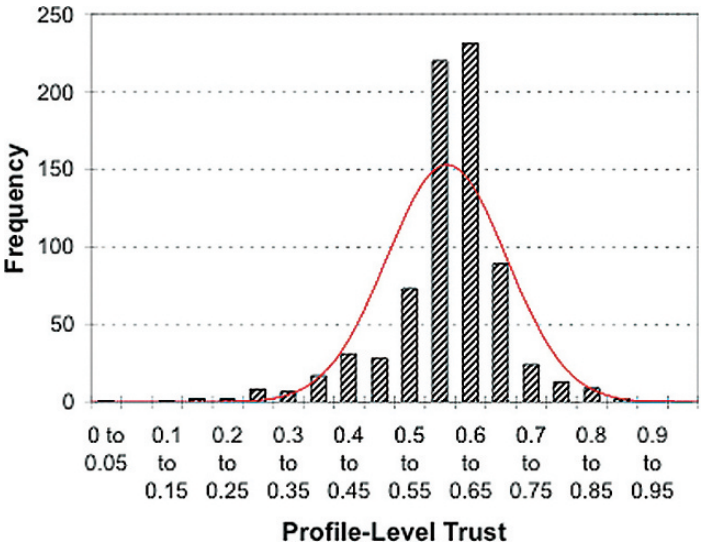

Fig. 9.6 The distribution of profile-level trust values among the producer profiles

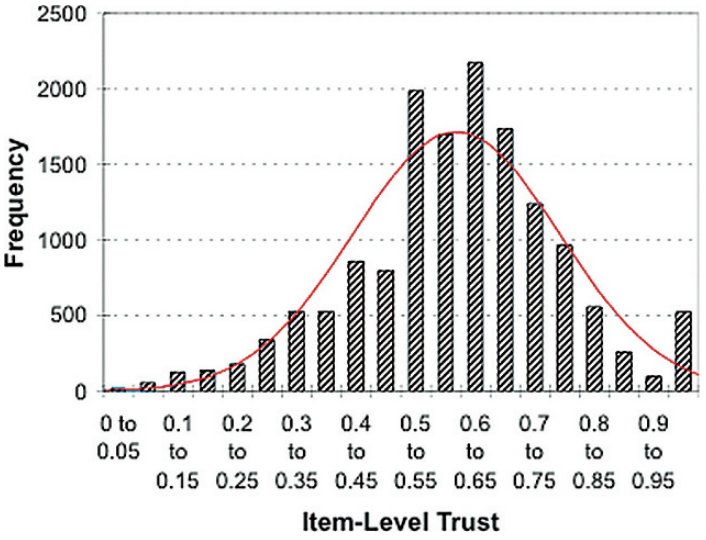

Fig. 9.7 The distribution of item-level trust values among the producer profiles

greater than 0.7. By comparison less than 4% of the profile-level trust values are less than 0.4 and less than 6% are greater than 0.7.

If there was little variation in trust then trust-based prediction strategies would not be expected to differ significantly from the standard Resnick method, but of course since there is much variation, especially in the item-level values, then

significant differences between the predictions made by Resnick and the predictions made by our alternative strategies are expected. Of course whether the trust-based predictions are demonstrably better remains to be seen.

9.4.6 Recommendation Error

Ultimately, we are interested in exploring how the use of trust estimates can make recommendation and ratings predictions more reliable and accurate. In this experiment we focus on the mean recommendation error generated by each of the recommendation strategies over the items contained within the consumer profiles. That is, for each consumer profile, each of its rated items are temporarily removed and the producer profiles are used to generate a predicted rating for this target item according to one of the 7 recommendation strategies proposed above. It is worth pointing out that the rating error is calculated with reference to the item's known rating and an average error is calculated for each strategy.

The results are presented in Fig. 9.8 as a bar-chart of average error values for each of the 7 strategies. In addition, the line graph represents the relative error reduction enjoyed by each strategy, compared to the Resnick benchmark. A number of patterns emerge with respect to the errors. First, the trust-based methods all produce lower errors than the Resnick approach (and all of these reductions are statistically significant at the 95% confidence level) with the best performer being the combined

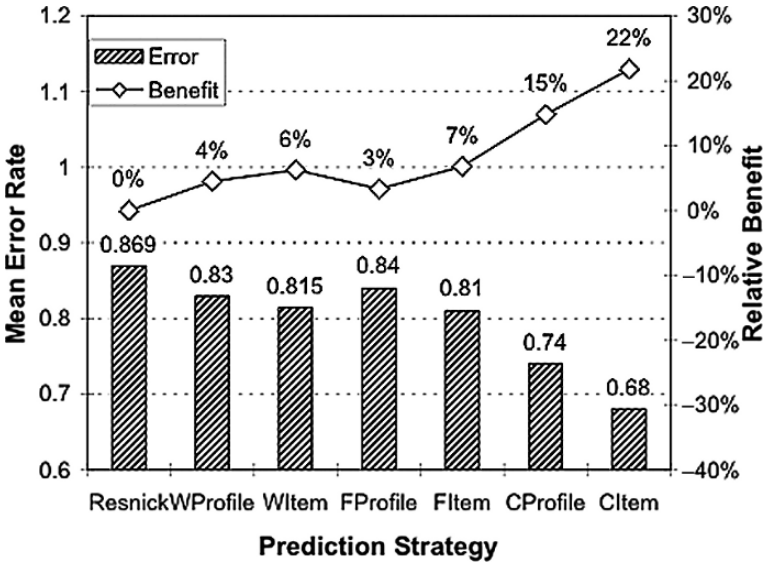

Fig. 9.8 Average prediction error and relative benefit (compared to resnick) for each of the trust-based recommendation strategies

item-level trust approach ($CItem$) with an average error of 0.68, a 22% reduction in the Resnick error.

It was found also that in general the item-level trust approaches perform better than the profile-level approaches. For example, $WItem$, $FItem$ and $CItem$ all out-perform their corresponding profile-level strategies ($WProfile$, $FProfile$ and $CProfile$). This is to be expected as the item-level trust values provide a far more fine-grained and accurate account of the reliability of a profile during recommendation and prediction. An individual profile may be very trustworthy when it comes to predicting the ratings of some of its items, but less so for others. This distinction is lost in the averaging process that is used to derive single profile-level trust values, which explains the difference in rating errors.

In addition, the combined strategies significantly out-perform their corresponding weighting and filtering strategies. Neither the filtering or weighting strategies on their own are sufficient to deliver the major benefits of the combination strategies. But together the combination of filtering out untrustworthy profiles and the use of trust values during the ratings prediction results in a significant reduction in error. For example, the combined item-level strategy achieves a further 16% error reduction compared to the weighted or filter-based item-level strategies, and the combined profile-level strategy achieves a further 11% error reduction compared to the weighted or filter-based profile-level approaches.

On average, over a large number of predictions, the trust-based predictions techniques achieve a lower overall error than Resnick. It is not clear, however, whether these lower errors arise out of a general improvement by the trust-based techniques over the majority of individual predictions, when compared to Resnick, or whether they arise because of a small number of very low error predictions that serve to mask less impressive performance at the level of individual predictions. To test this, the percentage of predictions where each of the trust-based methods wins over Resnick are examined here, in the sense that they achieve lower error predictions on a prediction by prediction basis.

These results are presented in Fig. 9.9 and they are revealing in a number of respects. For a start, even though the two weighting-based strategies ($WProfile$ and $WItem$) deliver an improved prediction error than Resnick, albeit a marginal improvement, they only win in 31.5% and 45.9% of the prediction trials, respectively. In other words, Resnick delivers a better prediction the majority of times. The filter-based ($FProfile$ and $FItem$) and combination strategies ($CProfile$ and $CItem$) offer much better performance. All of these strategies win on the majority of trials with $FProfile$ and $CItem$ winning in 70% and 67% of predictions, respectively.

Interestingly, the $FProfile$ strategy offers the best overall improvement in terms of its percentage wins over Resnick, even though on average it offers only a 3% mean error reduction compared to Resnick. So even though $FProfile$ delivers a lower error prediction than Resnick nearly 70% of the time, these improvements are relatively minor. In contrast, the $CItem$, which beats Resnick 67% of the time, does so on the basis of a much more impressive overall error reduction of 22%.

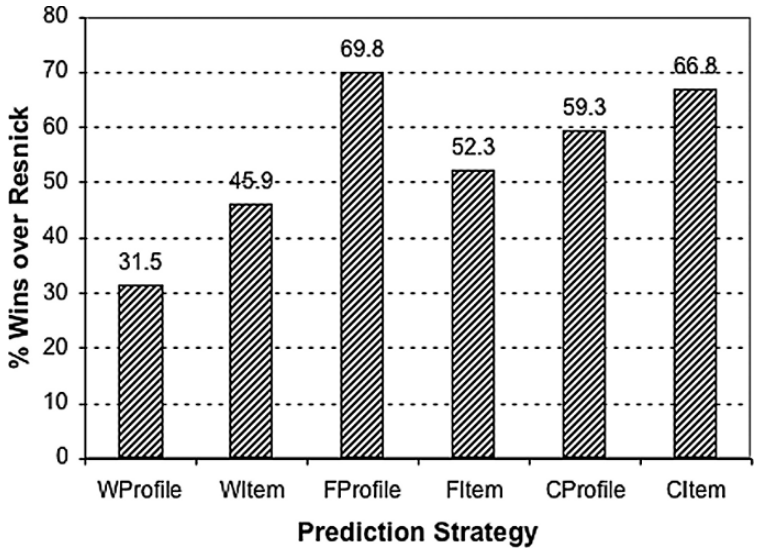

Fig. 9.9 The percentages of predictions where each of the trust-based techniques achieves a lower Error prediction than the Benchmark Resnick technique

9.4.7 Discussion

This concludes the implementation and evaluation of the first of the three techniques for sourcing trust in Social Web applications. The trust model presented used *indirect* trust, since it computes trust between users using their ratings on items as a proxy. The trust source in the model was *implicit*, since the users were not knowingly providing trust information while they were making their ratings. The following section presents the second technique for sourcing trust, this time in the domain of online auctions. In this technique, trust values are computed from negative sentiment which can be discovered from feedback comments existing in the system. This trust source can be classified as *direct*, since the users are expressing their views directly about other users in their comments. The trust values can be also considered as *attentive*, since users are knowingly expressing trust for other users as they enter feedback on the system.

9.5 Source 2: Extracting Trust From Online Auction Feedback Comments

Online auctions and marketplaces where users leave text based feedback comprise a significant percentage of Social Web applications. The second technique for modelling trust in Social Web focuses in this area. To address the problem of unnaturally

high trust ratings on trading communities such as eBay, we look to the freetext comments and apply a classification algorithm tailored for capturing subtle indications of negativity in those comments. The situation arises frequently where users are afraid to leave a negative comment for fear of retaliatory feedback comments which could damage their own reputation [19]. In many of these cases, a positive feedback rating is made, but the commenter still voices some grievance in the freetext comment. This is the type of subtle but important information the *AuctionRules* algorithm attempts to extract.

9.5.1 The AuctionRules Algorithm

AuctionRules is a classification algorithm with the goal of correctly classifying online auction comments into positive or negative according to a threshold. *AuctionRules* capitalises on the restrictive nature of online markets: there are a limited number of salient factors that a user (buyer or seller) is concerned about. This is reflected in feedback comments. We define a set of seven core *feature sets* for which the algorithm will compute granular trust scores. The following sets have a coverage of 62% of the comments in our database. The algorithm can obtain semantic information from 62% of the comments at a fine grained level. It is shown in our experimental analysis how we can maintain over 90% coverage using this algorithm. The terms in brackets are contents of each feature set.

1. *Item* - The quality/condition of the product being bought or sold. (*item, product*)
2. *Person* - The person the user makes the transaction with. (*buyer, seller, eBayer*, *dealer*)
3. *Cost* - Cost of item, cost of shipping, hidden costs etc. (*expense, cost*)
4. *Shipping* - Delivery of the item, security, time etc. (*delivery, shipping*)
5. *Response* - Communication with the other party, emails, feedback comment responses. (*response, comment, email, communication*)
6. *Packaging* - The packaging quality/condition of the item (*packaging*)
7. *Payment* - how the payment will be made to the seller, or back to buyer for return (*payment*)
8. *Transaction* - the overall transaction quality (*service, transaction, business*)

This technique enables us not only to compute a personal trust score between individual users, but also to provide more granular information on a potential transactor. For example: 'User x is very trustworthy when it comes to payment, but shipping has been unsatisfactory in the past', This granular or *contextual* trust draws on the wealth of information in comments and can uncover hidden problems, which the current trust system on eBay might overlook. For example, if John in New York wants to order a crate of Italian wine from Lucia in Italy, it would be very beneficial for John to examine Lucia's history of shipping quality.

Figure 9.10 details *AuctionRules* working on a sample comment, which had a positive rating on eBay. (All of the explanation in this section refers to Fig. 9.10.)

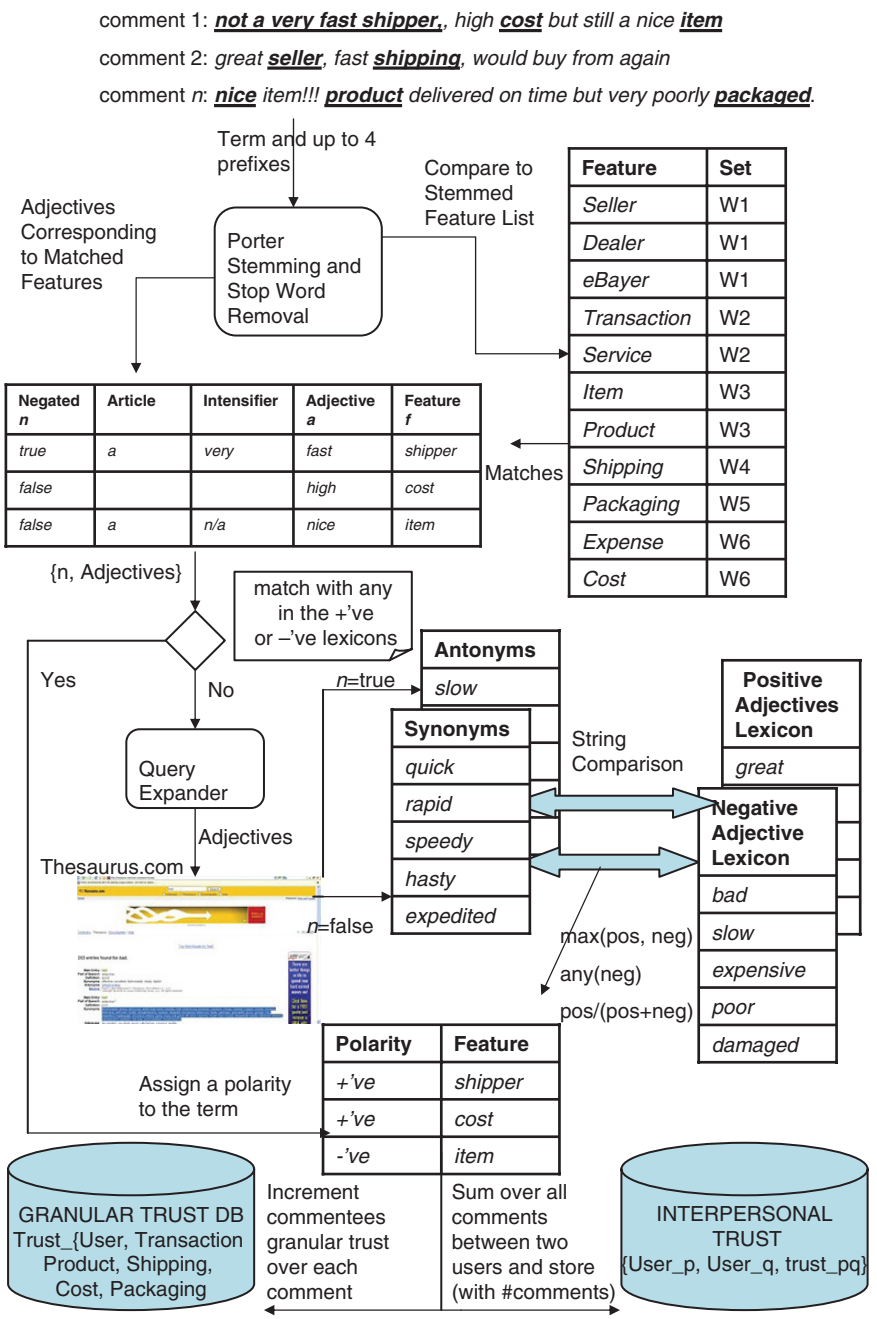

Fig. 9.10 The *AuctionRules* comment classification algorithm

Each term in a comment and up to four preceding terms are passed into an implementation of the Porter stemming algorithm [17]. The standard porter stemmer uses many rules for removing suffixes. For example, all of the terms in c are conflated to the root term *'connect'*. c = *connect*, *connected*, *connecting*, *connection*, *connections*. This reduces the number of terms and therefore the complexity of the data. The stemmer algorithm was modified to also stem characters not used by *AuctionRules*, such as '?, !, *, (,)' for example.

Data dimension is reduced by removal of stop-words. Google's stop-word list[2] was used as the removal key. A future addition to the algorithm would put an automatic spelling corrector at this point. There were 11 different spelling occurrences of the word 'packaging' for example. Each stemmed term is compared against the stemmed terms from the feature list. If a match is found, the algorithm examines the preceding term types. This is shown graphically in Fig. 9.10. The algorithm recognises five term types:

- *nouns* the words contained in the feature sets.
- *adjectives* (e.g: 'nice', 'good') from a Web list of 150
- *intensifiers* (e.g 'very', 'more') list of 40.
- *articles* 'a' or 'the'
- *negators* (e.g. not, 'anything but') from a manually generated list (currently 17).

The table on the left in Fig. 9.10 shows the order in which these terms are analysed. From the five terms, two can provide semantic meaning about the feature: adjectives and negators. If an adjective is found without a negator, it is compared to an arrays 20 positive and an array of 20 negative adjectives. If a negator is found as the second or third preceding term, the process works with positive and negative arrays switched. If a match is found, the polarity for that feature is recorded.

If no match is found, *AuctionRules* uses query expansion, by calling an interface to an online thesaurus which returns an array of 20 synonyms. If a negator is present, the interface returns an array of 20 antonyms, and these lists are compared in a similar manner to our short lexicon of positive and negative adjectives. The matching results are recorded in three ways: (a) $max(pos, neg)$ (b) $any(neg)$ and (c) $neg/pos + neg$. In the case of (c) the polarity is recorded according to a configurable threshold α. Two separate trust databases are maintained: *granular* or *contextual* trust which is the trust computed for each feature for a user over all of the comments analysed. Equation 9.13 shows contextual trust t as a multi valued set of trust scores associated with each feature. Here, f denotes a particular feature and t_fn is the associated trust score. The second trust database is *interpersonal* trust which is the average trust value recorded for all features on every comment made between two users.

$$t_{\text{granular}} \varepsilon \{t_{f1}, t_{f2}, \ldots t_{fn}\} \tag{9.13}$$

[2] http://www.ranks.nl/tools/stopwords.html

Of course not every comment will contain words from the feature lists above. In cases where no features are found, the algorithm performs a straightforward count of positive and negative terms using the query expansion module where necessary. In this manner, coverage is maintained at over 90%, and many of the unclassifiable comments are foreign language or spelling mistakes. For example, from all the (10k) eBay comments stored by our crawler, 68 different misspellings of the word 'packaging' were found. It would be interesting as future work to analyse the effects of automatic spelling correction on our results.

One drawback with the current implementation is that negative sentiment which is represented in complicated grammatical expressions is often misinterpreted by the algorithm. For example, using the current implementation the sentence 'the item was anything but good, shipping was ok' gets misinterpreted to 'good shipping'. To overcome this more complex natural language processing rules need to be developed to account for more complex statements. However, the vast majority of comments crawled from eBay are expressed in simple grammar, so instances of the above problem are relatively few. There are several available 'black box' NLP classifiers, for example, the Stanford university NLP tools.[3] Future work on this algorithm should include a comparison between *AuctionRules* and these NLP classifiers.

9.6 Evaluation

We examine four factors in our evaluation of *AuctionRules*, including the *accuracy* of the *AuctionRules* classifier with respect to other techniques from machine learning. We also examine accuracy from a Mean Absolute Error perspective and by creating confusion matrices to examine performance with respect to false negative and positive classifications. As the system uses a very small amount of domain knowledge, and a limited number of features, we must examine the *coverage* of *AuctionRules* over our comments database. Finally, we make a comparison between the *scale* of trust achieved by *AuctionRules* against the current eBay scale.

9.6.1 Setup

Figure 9.11 explains the environment in which we test our algorithm. Data is crawled from the online auction site according to the crawler algorithm below. Importantly, unlike it's machine learning counterparts, *AuctionRules* requires *no* knowledge of the comment set it has to classify. The feature lists used by the algorithm are generic and should work on any set of online auction comments. Arguments have been made that *AuctionRules* relies heavily on domain knowledge and that this is very restrictive. The *AuctionRules* algorithm uses little domain knowledge in the form of lexicons of positive and negative words, intensifiers and so on.

[3] http://www-nlp.stanford.edu/software/index.shtml

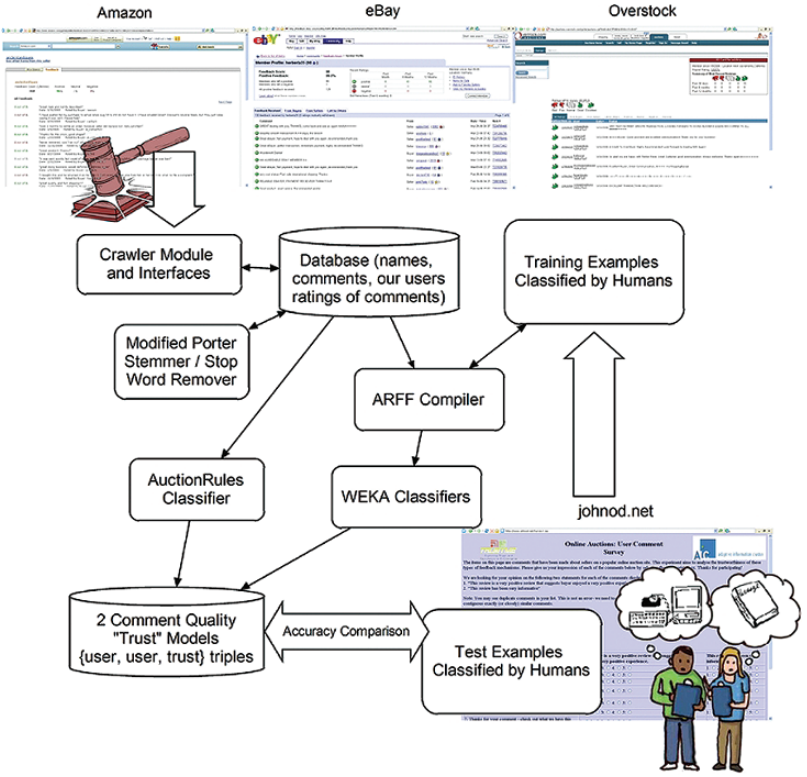

Fig. 9.11 Graphical overview of the trust-modelling process. (Current implementation only uses eBay as a source for ratings.)

AuctionRules will work on any system where users buy and sell from each other and leave textual feedback comments using no more domain knowledge than is used in the eBay system. Relative to the size of the application domain, the knowledge required by the algorithm is very minimal.

Algorithm 9.6.1: CRAWL($String\ url\_list,\ int\ maxbound$)

while $n < maxbound$
for $i \leftarrow 1$ **to** $\#Items\_on\_page$
do $\begin{cases} followSellerLink(); \\ \textbf{for } j \leftarrow 1 \textbf{ to } \#Comments\_on\_page \\ \quad \textbf{do } db.add(bId, sId, comment, sTrust, bTrust); \\ n \leftarrow n + 1; \end{cases}$
return ;

10,000 user comments were collected from the auction site using the crawler. As a social network visualisation was planned, data was collected from within a

domain with highly interconnected and contained repeat-purchase users. After a manual analysis of a range of sub-domains of the auction site, Egyptian antiques was chosen as the data domain as it appeared to meet the prerequisites to a reasonable degree. Although a large number of comments were collected, only 1,000 were used in our experimental analysis.

9.6.1.1 User-Provided Training Data

In order to test any comment classification algorithm a benchmark set was required. 1,000 comments were compiled into an online survey[4] and rated by real users. In this survey, users were asked to rate the positiveness of each comment on a Likert scale of 1–5. 10 comments were presented to a user in each session. Each comment was made by different buyers about one seller. Users were required to answer the following:

- How positive is the comment (Average rating: 3.8442)
- How informative is the comment (Average rating: 3.1377)
- Would you buy from this seller (Average rating: 4.0819)

Figure 9.12 shows a screenshot of the comment classification pages where users provided ratings on each of the comments in our database. Currently, only results from the first question are used to develop and test *AuctionRules*. For future experiments we may incorporate results from the other questions. Permission was sought from eBay inc. to use the information from the eBay Web site in our experiments.

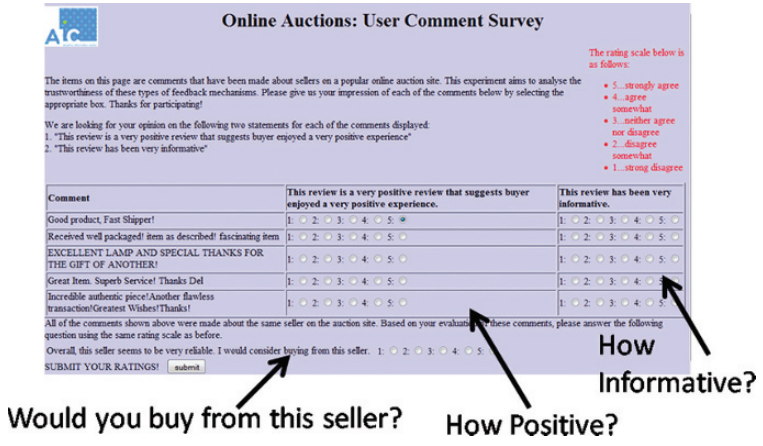

Fig. 9.12 Screenshot of Web-based Comment Rating Pages where users provided opinions on each crawled comment

[4] www.johnod.net/Surveyone.jsp

9.6.2 Comparing AuctionRules With Machine Learning Techniques

To examine classification accuracy of *AuctionRules*, it was tested against 7 popular algorithms. We chose three rule-based learners, *Zero-r*, *One-r*, *Decision Table*, one tree learner *C4.5 rules*, two Bayes learners, *Naive Bayes* and *BayesNet* and a lazy learning algorithm *K-Star*.

Figure 9.13 shows results of this experiment. For each algorithm we performed three runs. a 60:40 train-test split, an 80:20 split, and a 10-fold cross validation of the training set, which randomly selects a training set from the data over 10 runs of the classifier and averages the result. In the experiment, each algorithm made a prediction for every value in the test set, and this prediction was compared against the training set. *AuctionRules* beat all of the other classifiers in *every* test we performed, achieving over 90% accuracy in all of the evaluations, 97.5% in the 80:20 test, beating the worst performer *K-Star* by 17.5%, (relative 21.2%) and it's closest competitor *Naive Bayes* by 10.5%, giving a relative accuracy increase of 12.7%.

In addition to numerical accuracy, we examined where the high accuracy results were coming from more closely by assessing the confusion matrix output by the algorithms. This was necessary since prediction of false negatives would have an adverse effect on the resulting trust graph. This phenomenon has been discussed by Massa in [13] with respect to the *Moleskiing* application, and Golbeck in [6] with respect to the TrustMail application. Table 9.2 shows *AuctionRules* outperforming all of the other algorithms by predicting no false negatives. This is taken as a good result since in a propagation computation negative values should contain more weight because of their infrequency. When a value is presented to a user directly however, false positives are more damaging for an auction environment. *AuctionRules*also performs very well for false positives with a rate of 4.5%, half

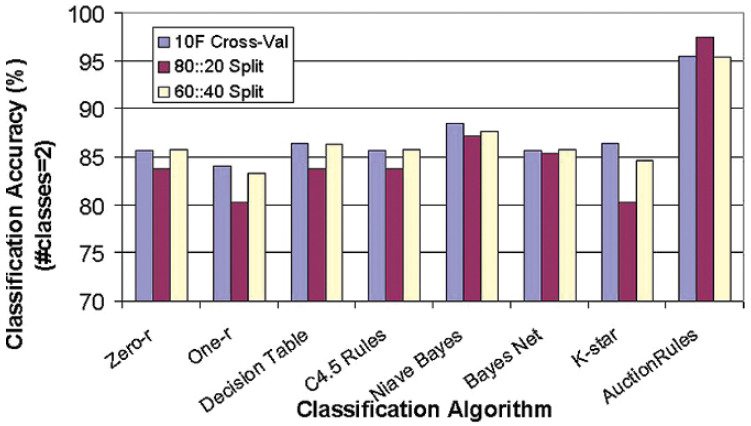

Fig. 9.13 Classification Accuracy [Classification Distribution from User Evaluations: 36% positive, 63% negative, using a threshold of 4 or higher for a positive comment.]

Table 9.2 Confusion matrices showing percentage true and false negatives and positives for four of the algorithms tested. [All of the other algorithms had similar results to the ones displayed.]

	AuctionRules		*NaiveBayes*		*Decision Table*		*One-r*	
	+'ve	**-'ve**	+'ve	**-'ve**	+'ve	**-'ve**	+'ve	**-'ve**
+'ve	91.4	*0*	84.1	1.2	84.6	1.2	77.3	8.1
-'ve	4.7	4.7	11.1	2.9	12.3	1.7	8.5	5.9

that of the closest competitor *One-r*. All of the algorithms displayed similar trend to the ones in Table 9.2, which shows results of the 80:20 classification experiment which had a test set of 234 comments. It was found during accuracy evaluations that there was a strong correlation between the number of feature terms recognised and the final accuracy of the classification. For our coverage experiments, we addressed the number of *hits* found with respect to coverage. This is detailed in the following section.

9.6.3 Coverage and Distribution Experiments

To assess the coverage of the *AuctionRules* feature-based trust calculator we examined the number of feature term hits that occur during classification. Coverage was tested in two modes. Firstly, the standard feature-based mode. In this case, 62% of the 1000 comments tested contained at least one hit from the feature list. This provides us with semantic knowledge about a large number of comments. However there is a sharp reduction in coverage when we look for comments with more than one hit. To increase coverage, *AuctionRules* uses simple term counting to supplement its feature-based approach in cases where it fails to find any terms. When

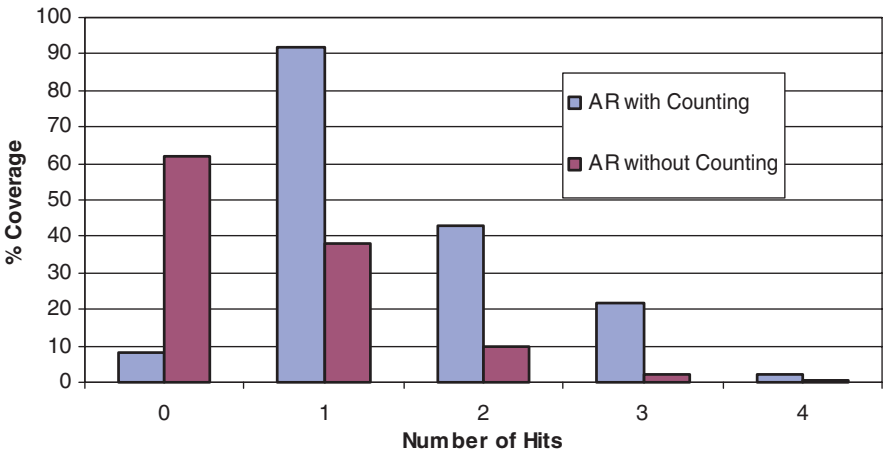

Fig. 9.14 Comparison of the coverage and hit ratio in *AuctionRules*

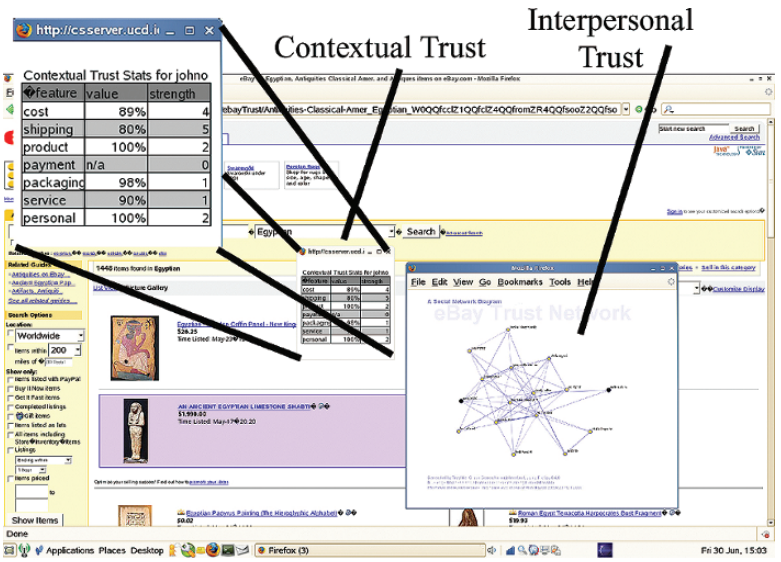

Fig. 9.15 Screenshot of the prototype auction interface showing contextual trust and interpersonal trust, mined from feedback comments

counting is applied the coverage is greatly increased with over 90% of the comments getting at least one hit. After manual examination, it is clear that majority of the other 8% can be attributed to misspellings, foreign language and other noise.

A distribution analysis was performed between the *AuctionRules* generated trust values and the current and found that the new trust values do capture some unseen negativity from feedback comments. This is manifested in a slightly more scaled distribution relative to the current eBay values. From 600 comments that were analyzed in the distribution experiment, 90 comments, which had been rated as 93–100 % positive, were taken out of that bracket by *AuctionRules*. This may seem like a small amount, but compared with the current system which has less than 1% negative ratings, *AuctionRules* produced 16%.

9.7 Discussion

The main contribution of this section is an algorithm for extracting *personalised* and *contextual* trust from the wealth of freetext comments on online auction sites. The algorithm operates on the assumption that online auction transactions can be categorised into a relatively small set of features. *AuctionRules* can extract context-specific and personal trust both retroactively and on the fly as new comments are added. There are a range of uses for the extracted trust scores. In this chapter we have shown one such use in a pop-up visualisation of the resulting trust network for a demonstration subset of the eBay marketplace. This visualisation also shows

per-feature trust in a pop-up window for a buyer or seller. In our evaluations we show that even using a small lexicon of key features, coverage is still maintained above 90%. We show that the *AuctionRules* classifier beats seven popular learning algorithms at this classification task by up to 21% in accuracy tests using very minimal domain knowledge. *AuctionRules* has a favourable false negative rate for the classifier of 0% compared to up to 6% from benchmark algorithms and a false positive rate of 4.7% compared with 8.5–12.3% for the benchmark algorithms.

In the context of the classification of trust sources presented earlier, the trust values computed by *AuctionRules* are *direct* and *implicit* because a transactor is knowingly expressing trust information to the system about another specific user. The following section presents the implementation and analysis of the last technique for sourcing trust- asking the user directly at the moment a trust value is required. An interactive interface for collaborative filtering systems is presented which allows users to view pre-computed trust values, and make modifications to them as they receive recommendations from the system. With respect to the classification scheme, the trust sources in this system are a combination of several types and can be classified as both *direct* and *indirect* since trust values between a user seeking recommendations and each of his neighbours are displayed initially based on items which they both have rated. However, once a user manipulates this value through the interactive interface, they are providing a *direct* expression of trust.

9.8 Source 3: Extracting Trust through an Interactive Interface

The third application for computing and using trust information in the Social Web also focuses on ACF recommender systems, but from a very different perspective. In this technique, the focus is on extracting trust information directly from the end user at recommendation time. This is a form of *direct* and *attentive* trust as we are allowing an active user to knowingly provide trust information about other individuals. The idea behind recommender systems is to provide a user with a useful recommendation. Many recommender systems use item-based [20] or user-based [18] CF. CF models the social process of asking a friend for a recommendation. However, for people with no friends, or even bad friends, the social process obviously fails. In the CF literature, this failure has been termed the ‘grey-sheep’, ‘sparsity’ and ‘early-rater’ problems. In the real world it is difficult to choose who your peers will be, as modelled in most current CF systems. Imagine you want a movie recommendation and can choose a group of peers, in addition to your existing group. You might choose Stephen Spielberg, among others if your in the mood for a Sci-Fi flick. However, what if you told Mr Spielberg your top 10 movies, and he hated all of them. With this knowledge, would you still want his contributions?

In this technique, which follows from work in [16] we are interested in allowing a user to express trust values on a set of neighbours in order to allow them to benefit from improved recommendations that more accurately reflect their current preferences and mood. To achieve this the interface must convey meaningful

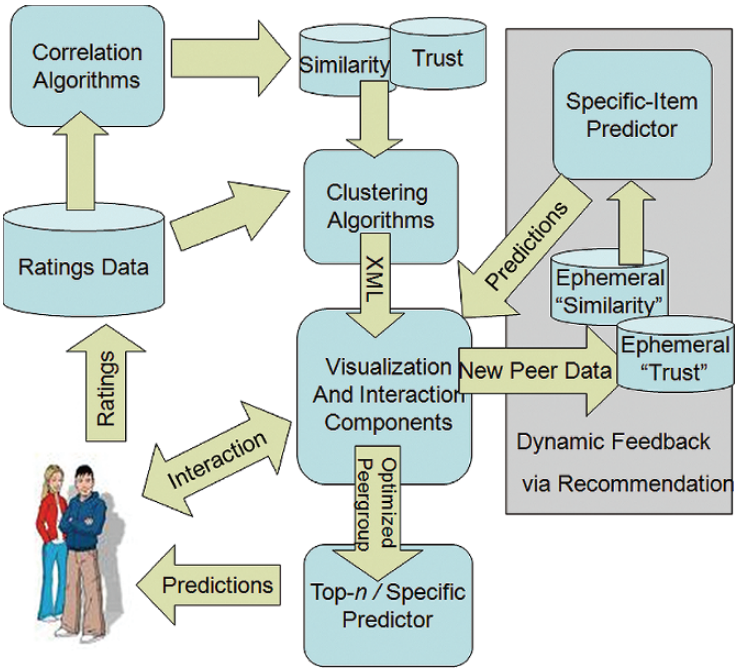

Fig. 9.16 Architecture of the PeerChooser visual interactive recommender system

information to the user about their neighbourhood structure and the interests of their neighbours. This is a challenging problem as each neighbourhood represents a high-dimensionality preference space which is difficult to render and even more difficult for a user to interpret. To address this, while maintaining ease of use for the end user, we have chosen to map the complex ratings data onto the space of movie *genres*, which are presented as a set of user-connected *genre nodes* alongside their neighbourhood representation; see Fig.9.17. An edge is created between a genre node g and a user u if u displays an affinity for g in their ratings; in practice we use a simple thresholding technique to identify affinities between users and genres if more than 10% of the user's positive ratings reflect a given genre. These genre nodes are also displayed using the force-directed graph layout algorithm, which causes a natural clustering of users to genres as shown in Fig. 9.17; in this example we allow each user to be connected to their two most dominant genres but in principle each user can be connected to any number of genres, subject to presentation clarity issues. *PeerChooser* provides highlighting functionality for nodes that are salient in the CF process. The k-closest nodes to the active user are highlighted in red. Any selected node is automatically scaled up to distinguish it from other nodes. Multiple node selection is enabled for group manipulation of neighbours. When a node is selected in *PeerChooser* the associated rating detail for that user is shown in a table. Genre nodes in *PeerChooser* are linked to an icon database to represent each genre clearly.

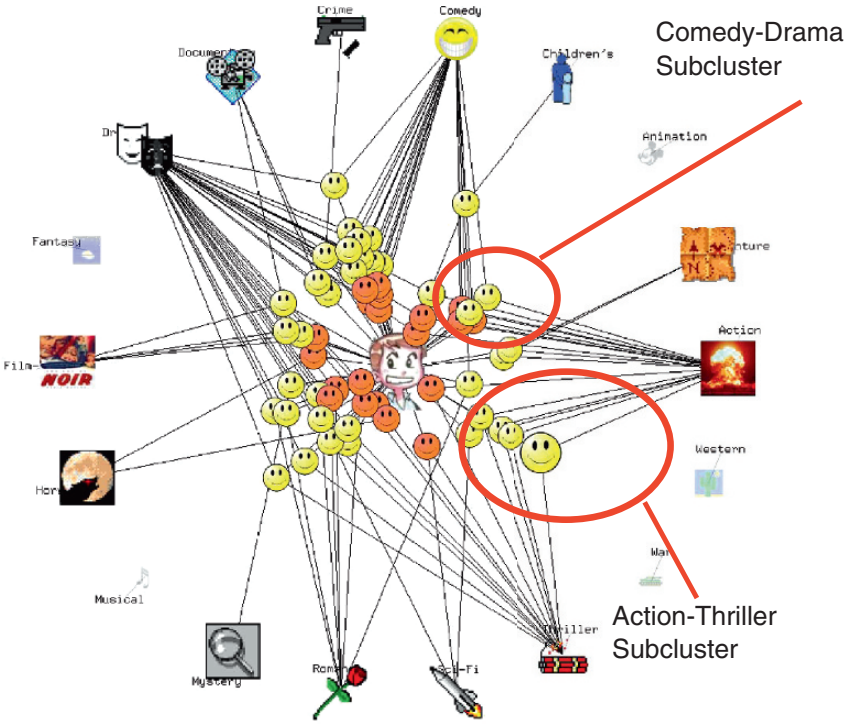

Fig. 9.17 MovieLens data visualised using *PeerChooser* with multiple genre associations

Labelling in *PeerChooser* is done on a per node basis. The label-positioning algorithm ensures that labels will not overlap on the graph and it attempts to place labels towards the outer edges of the graph wherever possible. Labels can be enabled or disabled on the basis of node type. For most of the visualisations in this paper we have displayed labels only on the genre nodes.

PeerChooser uses OpenGL technology on a Java platform, making it visually appealing to the end user. The layout algorithm is a force-directed graph which uses a standard spring model, with restrictions on the genre nodes. When the spring forces come to rest, the active user can interact with the graph by moving or deleting icons. Genre nodes are positioned around the outside of the graph and can be clicked on and moved freely. When this occurs, each connected neighbour node is moved by a relative distance in the same direction. This allows the user to make a bold statement about her current opinion on a particular genre. For example, if the comedy icon is moved towards the active user's avatar then all users who like comedy will be drawn closer and become 'more similar' to the active user for this session. We term this new value *ephemeral similarity*. Furthermore, users can make fine-grained statements by moving neighbour icons individually. On mouseover, nodes are highlighted and associated ratings are displayed in the right panel, shown in Fig. 9.18.

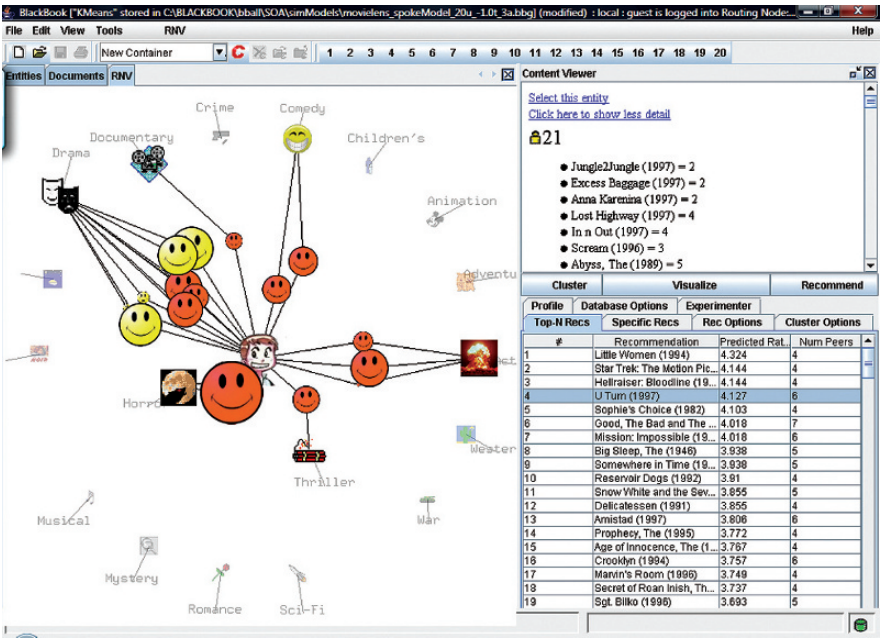

Fig. 9.18 The *PeerChooser* OpenGL application showing trust values as node size and correlation as edge length

9.8.1 Fair Representation of Genre Information

The genre distribution in the MovieLens data [citealpch09:bib15] has a massive bias towards three genres: Drama, Action, and Comedy. As a result, our initial clustered graphs tended to only show connections between non-active user nodes and nodes representing these three genres. To counter this problem and provide a more diverse set of neighbours we apply a scaling function to our edge drawing threshold based on number of occurrences of a genre in the database. Equation 9.14 shows how the system computes user to genre connectivity on a per-user basis. In Equation 9.14 G is the set of all genres, g represents one genre. Term $U_{\text{liked},g}$ indicates the number of items user U liked in genre g. U_{total} represents the total number of ratings by U; g_{total} is the number of movies in genre g and c is a scaling constant.

$$Max_{(g \in G)}\left(\frac{U_{\text{liked},g}}{U_{\text{total}}} + \frac{U_{\text{liked},g}}{g_{\text{total}}} \cdot c\right) \tag{9.14}$$

Once our data has been clustered based on the algorithm described in the previous section, the graph is saved as an XML file. The advantage of using this format is that graphs can be saved and loaded repeatedly, meaning that the personalised recommendation graph is portable between different deployments of *PeerChooser*.

9.8.2 Visualising Trust Relations in PeerChooser

In addition to providing an explanation of the recommendation process to the end user, *PeerChooser* enables the user not only to visualise *correlation*, but also the *trust-space* generated from the underlying ratings data.

Following from the trust-mining experiments in Chapters 4 and 5, the trust matrix built on the MovieLens dataset was incorporated into the visualisation mechanism as a means to provide at-a-glance information to the end user on *trust* in conjunction with *similarity*. For this experiment, trust was computed using the algorithms from Chapter 4, and similarity was computed in the usual way, using Pearson's correlation over the raw rating data.

Figure 9.18 shows the *PeerChooser* application displaying trust and correlation in parallel. In this personalised graph, the active user is positioned at the centre with a personalised avatar. Non-active nodes are positioned around this, again with edge length fixed proportional to the Pearson correlation between the users. Node size is a function of the trust- smaller icons are less trustworthy and larger ones have higher trust. Using this graph the active user can easily discern the differences between similar users and trustworthy users. In this example, a highly trusted neighbour can be seen just below the active user's avatar. This neighbour is also highly correlated due to the close proximity to the active user node. Attached to the trusted peer is an icon representing the 'horror' genre. In the right panel the top-*n* recommendation list contains the slightly esoteric movie 'Hellraiser' in the top three, with a prediction of 4.1 from 5. This is most likely the influence of the trusted and correlated peer who likes horror movies.

9.8.3 Implementation

Three different attempts were made to produce an interactive visualisation for ACF, firstly a 2-dimensional graphing tool was used to plot our data, and secondly a 3-dimensional approach was used. The graphs produced from these attempts are shown in Fig. 9.19 However, in the layout algorithms in both of these approaches introduced too much noise to accurately base ACF predictions on the laid-out graph of the underlying data. The current *PeerChooser* interface, which we call Recommender Network Viewer (RNV) was developed to address the problem of noisy graph layouts for collaborative filtering visualisation. See for example Fig. 9.18. The layout of this graph in terms of node types, edges, and their respective roles in the visualisation has already been discussed. Now we provide some detail on the underlying implementation of the dynamic graph in RNV.

9.8.3.1 Distance Weighting

To incorporate user hints into the recommendation process we simply replace the standard similarity values (based on user-user ratings correlations) with a new similarity value that is based on the inverse *Euclidean distance* between the active user

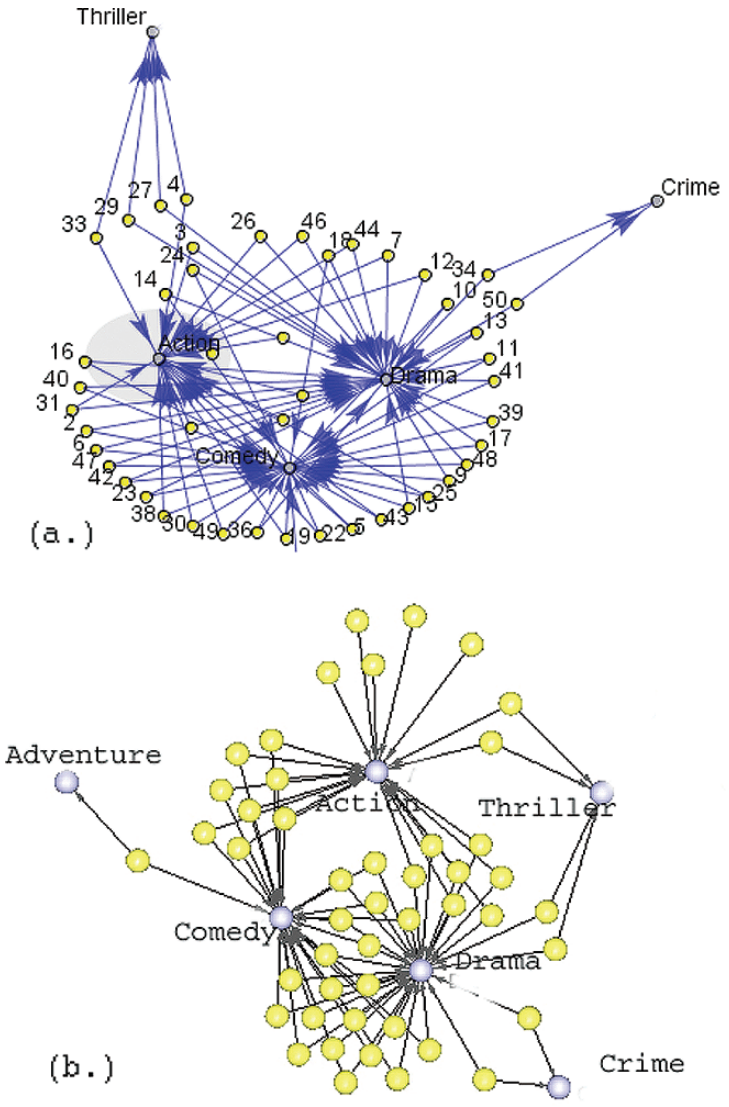

Fig. 9.19 MovieLens data visualised using SNV (2D) and DNV(3D) OpenGL-based visualisation tools

node and each of the k peer nodes that have been manipulated by the user. This is our *ephemeral similarity* value and is given by Equation 9.15. Here, Euclidean distance between pixels on the graph is normalised to the Pearson's correlation range of (-1, +1), max_dist is the maximum possible distance between the active user node and a peer node, while $node\_dist$ is the distance between the active node (i.e: the centre of the graph) and each peer node. Equation 9.16 shows the original Resnick prediction

formula using ephemeral similarity in place of the standard Pearson correlation. The nomenclature is similar to that in Equation 9.1 with $c(i)$ being the predicted rating for an active user c on item i.

$$\text{eph\_sim}(c, p) = 2(1 - \frac{\text{node\_dist}}{\text{max\_dist}}) - 1) \tag{9.15}$$

$$c(i) = \overline{c} + \frac{\sum_{p \in P(i)} (p(i) - \overline{p}) eph\_sim(c, p)}{\sum_{p \in P_i} eph\_sim(c, p)} \tag{9.16}$$

9.9 Evaluation

The majority of experiments involving recommender system algorithms are based on some form of automated testing, for example, predicting ratings for some 'hidden' subset of the rated data. This is only possible in absence of interactive components such as the ones of our *PeerChooser* system. Visualisation and interaction are additional 'tiers' to the process of collaborative filtering. The visualisation and interaction tiers bring many obvious advantages; however, it does prohibit the standard automated testing procedure since real people must explore and interact with the graph to attain results.

To evaluate the interactive visualisation components of the system, we performed controlled user trials. The objective of the trials was to ascertain as much information as possible about the users' overall experience using the visualisation techniques for profile generation and ratings prediction.

9.9.1 Experimental Data

For all of the evaluations in this chapter the small Movielens[5] [15] dataset was used. The base dataset consists of 100,000 ratings from 943 users on 1,682 items. The dataset has a sparsity of 93.7%.

The dataset contains no demographic information, but does contain genre information for each movie. Movies can be associated with multiple genres. Data is stored in a relational database and cached into memory where possible.

9.9.2 Rating Distributions

During user evaluations with our system two important comments were noted. Firstly, a 21 year old pointed out that he didn't know many of the movies in the

[5] http://movielens.umn.edu

dataset, and secondly, a 30 year old stated that there were a lot of 'classics' in the movie list he had to rate. These were worrying comments- it was feared that users might only remember the better, more popular movies, and accordingly produce an unnaturally high rating scale. To assess this possibility, we performed a comparative analysis between the distribution of dataset ratings and those from our user survey. Results from this analysis are presented in Fig. 9.20. There is a definite increase in the number of ratings in the top bin, but we were happy to find that the difference in the overall rating trend was not significant ($p<0.01$).

9.9.3 Procedure

Our study involved 30 users, ranging in age between 21 and 30 years. Each trial took approximately 30 mins and consisted of three stages, a pre-study questionnaire in which participants provided general demographic information, and information relating to their experience using recommender systems and graph visualisations. We also asked the users to rank a list of movie genres according to their perceived popularity.

The second stage of the user trial required the participant to use the system to get recommendations.

Firstly, users were required to perform a familiarisation task. This involved users getting a recommendation for a movie from the list, interacting with the various user

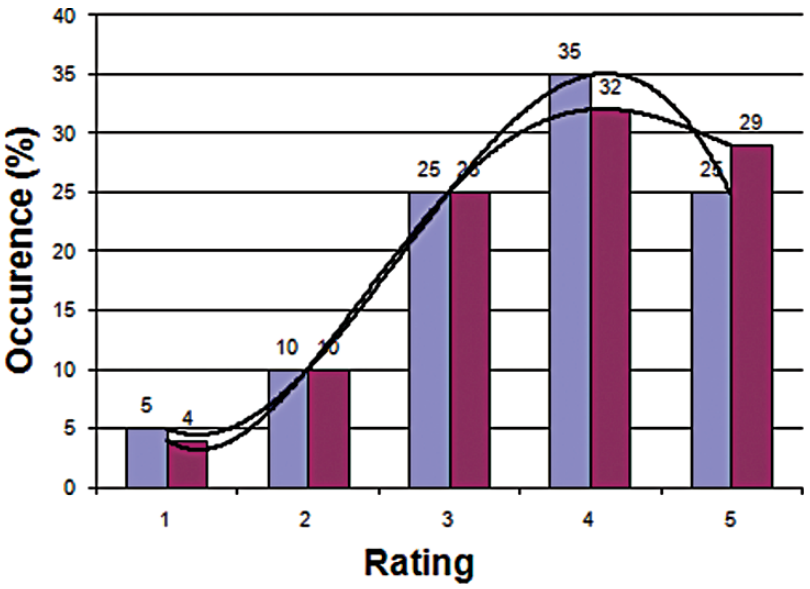

Fig. 9.20 Comparison of ratings distributions between existing MovieLens data and data from the user trials

nodes and genre nodes and manipulating the predicted rating to their taste. For the next task in the trial participants were required to use *PeerChooser's* manual rating window to generate a ratings profile via explicit ratings. Users were asked to rate 30 movies that they had watched on a scale of 1 (strong dislike) to 5 (strong like). Users then clicked on a submit button and the system correlated their ratings profile with the existing profiles.

To test the performance of our system we examined four techniques for generating recommendations, two with interaction and two without. For each approach we collected three values per item: the predicted rating, the actual rating, and the average rating in the whole database. The following list describes each approach:

1. *Average Layout* – The graph is laid out based on an 'average user'. We created this user by taking the average rating for the top 50 rated items. This technique was expected to yield the worst results as it contained no personalised information.
2. *Profile-Based Layout* – This is the benchmark CF algorithm. Correlations computed from a ratings profile are used to generate predictions.
3. *Profile-Based Layout with Manipulation* – Same as above but the user can manipulate the graph to provide information on current requirements.
4. *Profile-Based Layout with Manipulation and Feedback* – Same as above except the user receives dynamic recommendations on their salient items with each graph movement. We expected this to demonstrate the best performance.

For the *Average Layout* task users did not interact with the graph. They were simply shown a list of recommended items and asked to rate them (without showing them the predicted rating). Following from work by Swearingen et al. in [21] the predicted ratings were not displayed during this test as they could effect the users input. (Swearingen suggests that users tend to rate towards the machine-provided ratings.) Users then clicked on a submit button and the ratings were recorded.

A graph was then generated using our Pearson Clustering method described earlier. The cluster algorithm had a liked-item threshold of 3, meaning that the system only assumes an item is liked if the rating is 4 or 5. The number of neighbours to use in the computation was set to k=30 and the total number of users displayed on the graph was 400. The algorithm took the 400 most similar users from the entire database to display on the graph. The user then clicked on a button marked 'visualise' to show their personalised graph on the *PeerChooser* visualiser.

Users were then told to click on a button marked 'recommend' to a list of recommendations based on their personalised graph. Since the visualisation algorithm has zero effective noise for the collaborative filtering process, these recommendations are exactly equivalent to the benchmark Resnick CF algorithm from [18]. As with previous tasks, users rated 20 items and recorded their info. The penultimate recommendation task was similar to the above but with the addition of user manipulation on the graph.

In the final task the graph layout was as in the previous task. Users were asked to click checkboxes next to 5 movies they really liked and 5 that they hated. Users were

then told to click on a button marked 'specific predictions' and were provided with the benchmark CF predictions on those items. Users were then told to take some time to manipulate the graph to try and tweak the recommendations for their chosen items to a value that most suited them. All users in the survey reported that they had arrived at a satisfactory position within 2 mins without notification of a time constraint. At each movement users were provided with dynamic recommendations on their 'salient' item-set according to the current neighbourhood configuration. This allowed the users to dynamically assess the goodness of each movement performed. Users were again presented with a list of recommended movies and asked to rate them and store the information.

The concluding part of the user trial was the post-study questionnaire. This contained the important questions regarding the users overall experience with the interface. Results are shown in Fig. 9.22. Plotted are (as ratings between 1 and 5): Subjective accuracy of the different methods as perceived by the user (not based on actual prediction comparisons), the understandability of the used graph, user ratings for the four techniques in Fig. 9.21. The next question is important: Would the user prefer a visualisation-based approach over the standard active ratings-based technique in real life CF applications? The answer was a resounding 3.75 on average, one of the highest values in the survey. Likewise high ratings were given for the questions if the users gained knowledge through their interaction with the visualisation, the clarity of the labelling, informativeness of the right-hand information panel, desirability of applying this technique in other domains, and usefulness of interaction (highest value of all). Only on the question of how much control the users felt they had, and on the perceived accuracy did the survey yield ratings below average.

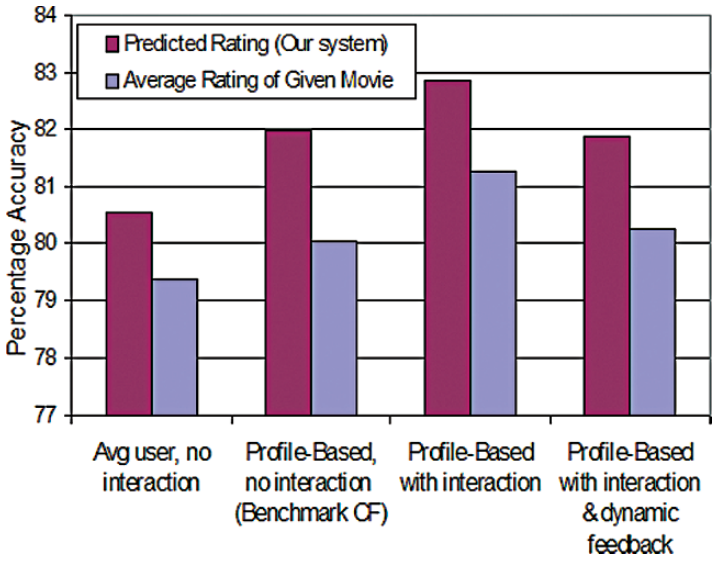

Fig. 9.21 Error results for each technique in the user trials

Table 9.3 Likert scale questions from the user survey

#	*QuestionDescription*
S1	I am familiar with recommender systems
S2	I am familiar with interactive computer graphics
S3	I am familiar with graph visualizations
S4	Which did you think was the most accurate
S5	I found the graph easy to understand
S6	Look at the 4 graphs and rate each one
S7	Did you prefer the visualisation approach
S8	I gained knowledge about the underlying data from the visualization
S9	I felt that the labelling was appropriate
S10	I felt that the information in the right panel was helpful
S11	I felt that the *visualisation* system gave me more control of the recommendation algorithm
S12	Would you like to see this interface on other domains (e.g: Amazon.com)
S13	I felt that the icons communicated the genres appropriately
S14	I felt that the *visualisation* system gave me more control of the recommendation algorithm
S15	I felt that I benefitted from *interaction* with the system

Also, during this final questionnaire users were asked once more to rate a list of popular genres, as in the pre study to assess if there is any learning effect from use of the visualisation, which turned out to be the case. As but one example, on the post-study form, every user listed Drama, the most popular genre in the list of 5 most popular genres. On the pre-study form, only 8 out of 13 users had mentioned Drama.

9.9.4 Recommendation Accuracy

To evaluate the accuracy of the techniques, mean absolute error was computed between the predicted rating and the users actual rating for each of the methods. Results of our analysis are presented in Fig. 9.21 for four techniques. As expected, the average predictions– that is, predictions based on the average user described earlier, exhibited the worst performance, producing a an accuracy of 80.5%. Predictions based on an average user (column 1 in Fig. 9.21) have high accuracy, this is not surprising if we take a look at the rating distribution graph in Fig. 9.20. Users tended to rated only movies they liked, and the average user was constructed from a list of the most commonly rated movies, which as it turns out were generally the most highly rated movies. This may be attributed to the fact that the movies were not new and users tended to remember them in a good light. Our profile-based technique (column 3) with manipulation beats the benchmark (column 2) achieving a small relative increase of 1.05%. A single factor between groups ANOVA shows that these differences are significant in each case with $p = 0.006$, $F = 3.87$. This small increase is an important result because it indicates that manipulation does increase recommendation accuracy. Future work includes conducting this experiment over an extended time period to gain a better assessment of the performance of our technique compared with a CF system operating solely on historical rating

data. We suspect that as data becomes more redundant, our method should become more accurate relative to standard techniques.

The most surprising result was that the dynamic feedback technique performed worse than the other profile-based techniques. After much analysis of the graphs it was determined that users tended to *over-tweak* the system to achieve desired results for their salient item sets. In doing this, many of the profile-based correlations were overwritten and the resulting layout was overfitted to the specific item sets. A solution to this may be to ensure diversity within the salient item sets.

To assess the effects of users interaction with the system a pre and post study questionnaire was answered by each participant. Table 9.1 lists each question from the survey and references the columns in Fig. 9.22. *S*1 to *S*3 indicate that all participants had experience with graphical interfaces, recommenders and visualisations. *S*4 tells us that, in contrast to our empirical accuracy tests, users felt that manual rating provided more accurate results. This was an interesting result which may indicate that users are more comfortable with familiar, manual rating systems. The four columns marked *S*6 represent participants' opinions on a range of different graph representations of the data, with column 8 being an exceptionally poor display. These are available for viewing at the URL above. *S*7 shows a clear preference for the visualisation approach, with 75% of users preferring *PeerChooser* over the traditional approach. The benefit of our technique as a recommendation *explanation* is shown by *S*8, where the majority of users felt they gained knowledge of the data from their interaction with the graphs. *S*9 and *S*10 indicate that users felt that labelling and node information were appropriate. *S*11 shows us that there were mixed views about the control that the interface provided on the CF algorithms. This response may be due to insufficient familiarisation time, since the technique does provide more access points to influence the CF process. *S*12 shows that more than 80% of participants agreed that they would like to see a *PeerChooser*-like interface

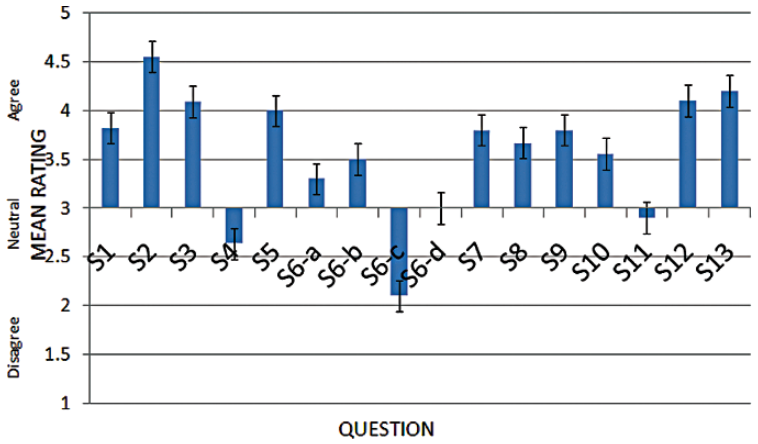

Fig. 9.22 Results of the questionnaire from the user trials

on other domains such as Amazon.com. This is an encouraging result, given that there is a broad scope of applications for our technique. More importantly, *S*15 shows that more than 80% of participants felt they benefitted overall from interacting with the system.

9.10 Comparison of Different Trust Sources

Trust information about users in the Social Web can be beneficial both to individuals and to the Social Web application as a whole. To illustrate the general benefits that are provided using each of the three trust-mining algorithms discussed in this chapter, a simple user-satisfaction trial was set up to ascertain the increase in user satisfaction, if any, created by integrating trust information from different sources back into each application. For this survey, 10 participants were asked to use each of the three applications twice. Once in normal operation, and once with the additional trust information added. For each application, participants overall satisfaction was elicited as follows:

1. *tRec*: Users rated 30 random items from the Movielens dataset. This information was used to compute recommendations using the item-level technique *CItem* to represent trust-based ACF. Users were also shown a list of 10 recommendations generated by the Resnick algorithm. (to represent normal ACF) Participants provided a percentage satisfaction with each recommendation and averages were taken for both modes.
2. *AuctionRules*: Users were asked to search for an item on a eBay-like interface. The first interface was similar to normal, showing sellers overall trust-ratings and feedback comments. The second interface, shown in Fig. 9.15 shows a trust-graph and contextual trust information mined from freetext comments for each seller in addition to the usual detail. For example, the general quality of packaging based on the information in the comments. For this experiment, users were simply asked to rate both presentation methods on a percentage scale. To account for the effects of ordering the interfaces were shown in varying orders. Averages were taken for both techniques.
3. *PeerChooser*: To analyse the effects of using trust information in the interactive recommender system on user satisfaction, participants were told to get 10 recommendations from the system, firstly using the visualisation technique but showing similarity values only, and secondly, showing both similarity and trust values. With trust as a function of node size and similarity as a function of edge length, as discussed earlier. Again, users rated their satisfaction with their recommended items on a percentage scale and an average value was taken for both modes.

Fig. 9.23 shows the results of this experiment for each of the three trust-mining algorithms presented. This graph plots the overall user satisfaction for each system,

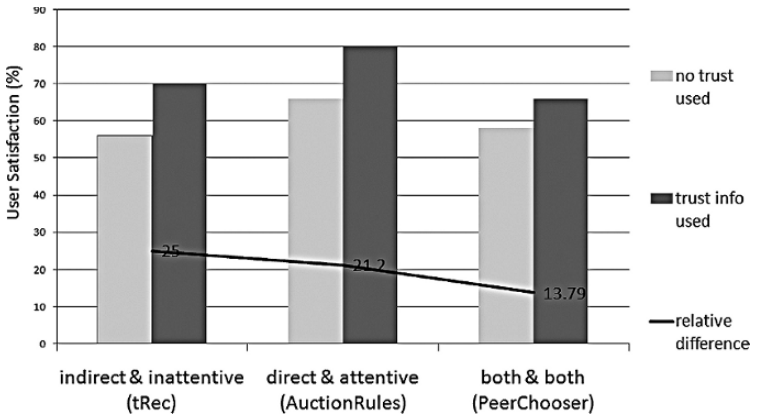

Fig. 9.23 Relative change in user satisfaction for each application operating with and without trust information

which harnessed its trust information from completely different sources. This graph is obviously not meant as a comparison between each technique, as comparison of user satisfaction across such heterogenous applications would not be viable. The important feature is the relative benefit score that is shown for each technique. This is the percentage increase in user satisfaction created by using trust data, over normal conditions (with no trust information) for each algorithm. For the *tRec* algorithm, which sourced trust information from *indirect* and *inattentive* data, (as discussed earlier), it is shown that using trust information can increase user satisfaction with predicted movie ratings by 30%, or a relative increase of 51%. It must be pointed out that this experiment is highly subjective and only computed over ten users. The motivation here is to show that an increase in user satisfaction can occur by using this trust source, more so than explicitly showing by how much.

For the *AuctionRules* algorithm, which sourced trust from freetext comments crawled from eBay, the average increase in satisfaction over the ten trials was 14%, or a relative increase of 21.2%. For *PeerChooser* the increase was 14%. Interestingly, the source of trust information for *AuctionRules* was both *direct* and *attentive*, meaning that users were expressing trust information directly about other users, and furthermore, they were aware that the they were expressing the information. Intuitively, this should have been the richest and most accurate source of trust information and probably should have resulted in more satisfied users than the other techniques, but it resulted in a smaller increase in user satisfaction, 3% less than the *tRec* algorithm, which used an *indirect* and *inattentive* trust source. *PeerChooser* uses both direct and indirect trust sources, since it uses the initial indirect values computed by tRec, then user-provided direct trust. This technique exhibited the smallest relative increase in user satisfaction with 13.9% However, there are a broad range of influencing factors and it is difficult to make an assumption as to why this was the case. What can be said about this data is that using additional trust

information about interacting users in Social Web applications tends to produce increases in their overall satisfaction with the system.

9.11 Conclusions

The Social Web can be thought of as a *participation platform* in which users perform both social and business interactions. As such, it brings about a need to assess the trustworthiness of people who a user may interact with. The core argument in this chapter is that trust can be sourced in a variety of ways on Social Web applications. Three techniques for harnessing trust from different sources within Social Web applications have been presented, focusing on two main application areas, namely recommender systems and online auctions.

The first application presented a technique for modelling trust in recommender systems based on users' history of contributions to the recommendation process. Based on the earlier classification, the source of trust information in this instance is both *indirect* and *inattentive* since the user is not rating other users directly, and is not necessarily aware that they are providing trust information. This trust model can be used to improve the accuracy of collaborative filtering recommendations by up to 22% over the benchmark Resnick prediction algorithm.

The second technique sources trust information from online auction feedback comments. To analyse the potential in this source, the *AuctionRules* algorithm was developed. This is a trust-mining algorithm for online auctions which harnesses the wealth of information hidden in online auction feedback comments to provide new information about the reputations of potential transactors. The *AuctionRules* trust model is based on lightweight natural language processing over free text comments. Based on our classification, this trust source can be considered *direct* since the comment is targeted at another specific user, and *attentive* since the user is aware they are linking themselves to this other user. Analysis presented shows that *AuctionRules* extracts useful trust information from comments on eBay, which can enhance the overall user experience with the system.

The final source of trust analysed is a direct link with the user. To facilitate and test trust information provided on-the-fly by users interacting with a Social Web system, the *PeerChooser* interface was developed. This is an interface for an ACF recommender system which enables users to pick and choose a peergroup for collaborative filtering during the recommendation process. The *PeerChooser* interface affords the user the opportunity to manipulate a visualisation of similarity and trust of their neighbours in parallel. Users can adjust these values as they see fit to represent facets of their current mood or requirements.

Each trust source presented in this chapter has been analysed in its own right, with live user evaluations and automated testing. It is shown that each technique that uses trust yields an increase in general user-satisfaction with their use of the Social Web application, relative to normal, non-trust-based operating conditions of the system.

References

1. Paolo Avesani, Paolo Massa, and Roberto Tiella. A trust-enhanced recommander system application. Molesking. In *SAC '05: Proceedings of the 2005 ACM symposium onApplied computing*, pages 1589–1593 ACM Press, New York, NY, USA, 2005.
2. Albert-Laszlo Barabási. *Linked: How E verthing Is Connected to Everything Else and What It Means for Business, Science, and Everyday Life.* Plume Books, April 2003.
3. Terry Bossomaier. Linked: The new science of networks by albert-laszlo barabási. *Artif. Life,* 11(3):401–402, 2005.
4. deSola Pool and Manfred Kochen. Contacts and influence. *Social Networks,* 1(1):5–51, 1978.
5. Malcolm Galdwell. *The Tipping Point: How Little Things Can Make a Big Difference.* Time Warner Books UK, January 2002.
6. Jennifer Ann Golbeck. *Computing and applying trust in Web-based social networks.* PhD thesis, Univeristy of Maryland at College Park, College Park, MD, USa, 2005. Chair-James Hendler.
7. Andun Josang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision.
8. Audun Jøsang, Elizabeth Gray, and Michael Kinateder. Analysing Topologies of Transitive Trust. In Theo Dimitrakos and Fabio Matrinelli, editors, *Proceedings of the First International Workshop on Formal Aspects in Security and Trust (FAST2003)*, pages 9-22, Pisa, Italy, September 2003.
9. Audan Jøsang, Elizabeth Gray, and Michael Kinanteder. *Simplification and Analysis of Transitive Trust Networks. Web Intelligence and Agent Systems: An International Journal*, pages 1-1, September 2005, ISBN ISSN: 1570-1263.
10. Cliff A.C. Lampe, Nicole Ellison, and Charles Steinfield. A familiar face(book): profile elements as signals in an online social network. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 435-444 ACM Press, New York, NY, USA, 2007.
11. S. Marsh. Formalising trust as a computational concept. *Ph.D. Thesis. Department of Mathematics and Computer Science,* University of Stirling, 1994.
12. Paolo Massa and Paolo Avesani. Trust-aware collaborative filtering for recommender systems. *Proceedings of International Conference on Cooperative Information Systems,* Agia Napa, Cyprus, 25 Oct 29, 2004.
13. Paolo Massa and Bobby Bhattacharjee. Using trust in recommender systems: an experimental analysis. *2nd International Conference on Trust Management, Oxford, England*, 2004.
14. S. Milgram. The small world problem. *Psychology Today*, 1:61-67, May 1967.
15. Bradley N. Miller, Istvan Albert, Shyong K. Lam, Joseph A. Konstan, and John Ridel. Movielens unplugged: experiences with an occasinally connected recommender systems. In *IUI '03: Proceedings of the 8th international conference on Intelligent user interfaces*, pages 263-266, ACM Press, New York, NY, USA, 2003.
16. John O'Donovan, Brynjar Gretatsson, Barry Symth, and Tobias Hollerer. Peerchooser: Visual interactive recommendation. *International Conference and Human Interaction (CHI'08)*, 2008.
17. M. F. Porter. An algorithm for suffix stirpping. *Readings in information retrieval*, pages 313-316, 1997.
18. Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Ridel. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceddings of ACM CSCW'94 Conference on Computer-Supported Cooperative Work*, pages 175-186, 1994.
19. Paul Resnick and Richard Zeckhauser. Trust among strangers in internet transactions: Empirical analysis of ebay's reputation system. *The Economics of the Internet and E-Commerce. Volume 11 of Advances in Applied Microeconomics.*, December 2002.
20. Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Reidl. Item-based collaborative filtering recommendation algorithms. In *World Wide Web*, pages 285-295, 2001.

21. Rashmi Sinha and Kirsten Swearingen. The role of transparency in recommender systems. In *CHI '02 extended abstracts on Human factors in computing systems*, pages 830-831. ACM Press, 2002.
22. Cai-Nicolas Ziegler and Georg Lausen. Propagation models for trust and distrust in social networks. *Information Systems Frontiers*, 7(4–5): 337-358, 2005.
23. Cai-Nicolas Ziegler and Michal Skubacz. Towards automated reputation and brand monitoring on the Web. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intleligence*, pages 1006-1070, IEEE Computer Society Press, Hong Kong, December 2006.