Collaborative Recommendation: A Robustness Analysis

MICHAEL O'MAHONY, NEIL HURLEY, NICHOLAS KUSHMERICK and GUÉNOLÉ SILVESTRE
University College Dublin

Collaborative recommendation has emerged as an effective technique for personalised information access. However, there has been relatively little theoretical analysis of the conditions under which the technique is effective. To explore this issue, we analyse the *robustness* of collaborative recommendation: the ability to make recommendations despite (possibly intentional) noisy product ratings. There are two aspects to robustness: recommendation *accuracy* and *stability*. We formalise recommendation accuracy in machine learning terms and develop theoretically justified models of accuracy. In addition, we present a framework to examine recommendation stability in the context of a widely-used collaborative filtering algorithm. For each case, we evaluate our analysis using several real-world data-sets. Our investigation is both practically relevant for enterprises wondering whether collaborative recommendation leaves their marketing operations open to attack, and theoretically interesting for the light it sheds on a comprehensive theory of collaborative recommendation.

Categories and Subject Descriptors: H.3.3 [**Information Search and Retrieval**]: Information Filtering

General Terms: Experimentation, Performance, Reliability

Additional Key Words and Phrases: Collaborative recommendation, machine learning, robustness

1. INTRODUCTION

Collaborative recommendation has emerged as an effective personalisation technique for a diverse array of electronic commerce and information access scenarios, e.g. Goldberg et al. [1992]; Shardanand and Maes [1994]. Such systems keep track of their customers' preferences, and use these data to offer new suggestions. Many variations have been explored, but the basic idea is as follows: to recommend items to a target customer, the system retrieves customers who have expressed similar preferences to the target, and then recommends items that were liked by the retrieved customers but not yet rated by the target.

The accuracy of various collaborative recommendation algorithms has been empirically validated for many domains, e.g. Breese et al. [1998], and the technology has been successfully deployed in many commercial settings. However, despite some

Authors' address: Department of Computer Science, University College Dublin, Belfield, Dublin 4, Ireland.

efforts [Kumar et al. 1998; Pennock et al. 2000; Azar et al. 2001; Drineas et al. 2002], there is no general theoretical explanation of the conditions under which a particular collaborative recommendation application will succeed or fail.

Our goal is to complement existing theoretical work by investigating the *robustness* of collaborative recommendation. In essence, robustness measures the ability of the algorithm to make good predictions in the presence of noisy data. Collaborative recommendation applications update their datasets when users enter new ratings, but there is no guarantee that these ratings reflect the user's true preferences. Indeed, since rating entry can be an onerous process, users may be careless and inaccuracies in the data must be expected.

More sinisterly, it is possible to imagine scenarios in which malicious users may be motivated to deliberately attack the recommendation system to cause it to malfunction. For example, publishers may wish to promote their work by encouraging a book recommendation system to output artificially high ratings for their publications. In some cases, the cost of perturbing the recommendation system's output may be prohibitive (e.g. only users who have purchased a book may be permitted to enter ratings). However, many collaborative recommenders are open Web services that malicious agents could easily attack. How much damage can they inflict?

There are two aspects to robustness. First, we may be concerned with recommendation *accuracy*: are the products recommended after the attack actually liked? The second issue is *stability*: does the system recommend different products after the attack (regardless of whether customers like them)?

While stability and accuracy are distinct, they are not independent. For example, if a recommender has perfect accuracy for a given task both with and without noise, then it must be perfectly stable. On other the hand, consider a product that no-one likes. The recommendation policies *recommend to no-one* and *recommend to everyone* are both perfectly stable, yet the first is always correct and the second is always wrong.

The remainder of this article is organised as follows. We begin with a discussion of collaborative recommendation and a formalisation of the notions of robustness and the perturbations with which we are concerned (Section 2). We then analyse robustness from both the accuracy and stability perspectives. Regarding accuracy, in Section 3 we formalise robustness in machine learnings terms, and introduce a novel form of class noise that models an interesting suite of attacks. We develop two models that predict the change in accuracy as a function of the number of fake ratings that have been inserted into the customer/product matrix. Regarding stability, we present a framework that describes the stability of a recommendation system subjected to various forms of attack (Section 4). In both cases, we empirically evaluate our predications against several real-world data-sets. We conclude with a description of related work (Section 3.4) and with a summary of our results and a discussion of several open issues (Section 5).

2. ROBUSTNESS AND COLLABORATIVE RECOMMENDATION

Fig. 1 illustrates the essential ideas of collaborative recommendation, as well as our concerns with robustness. Collaborative recommendation involves recording customer ratings for various products. For example, customer A likes product 1

but dislikes product 2.

Suppose the system wants to determine whether a particular target customer H will like product 7. To do so, the system would compare the ratings of customer H to all the other customers. For now, consider only the top portion of the matrix, customers A–H. The details depend on the particular similarity function employed, but presumably the system would determine that customers A and F are similar to H, since they tend to agree on the products they have rated in common. Recommender systems predict a rating for H based on the ratings given by the retrieved similar customers. In this case, we suppose that the system predicts that the target customer H will like product 7. Collaborative filtering research over the past decade has explored a wide variety of algorithms for accurately making such predictions.

Now suppose that a competitor to product 7 decides to influence the recommender system in order that product 7 is rarely recommended. To do so, a rogue competitor could pose as a set of fake customers I–M. What ratings should the competitor promulgate from these customers in order to decrease the rating for 7? (We call such a maneuver a *nuke* of product 7.) Clearly, none of the fake users should like product 7; see the last column of rows I–M. In this article we describe several ways in which ratings for products 1–6 might be generated. For now, the point is simply that the rogue profiles may well influence the recommender system's prediction. For example, rogue user L is now highly similar to the target customer, which may well lead the recommender system to switch its prediction from like to dislike.

This lack of robustness can be quantified in two ways. On the one hand, we can measure the accuracy of the predicted ratings: does customer H in fact like product 7? On the other hand, we can measure the stability of the system. Even if we can not determine whether customer H likes product 7, the mere fact that the recommender system switched its prediction may be problematic—even if the unbiased prediction was wrong!

In the limit, a malicious agent can clearly force a recommender system to behave in an arbitrary fashion, by adding to the ratings matrix enough rogue users to completely dilute the preferences of the real users. But such an attack would likely be extremely expensive, even assuming an open system where network bandwidth is the dominant cost, given the very large quantity of data that would need to be inserted to achieve the desired outcome. Thus, any potential gain in market share or whatever other factors lead the malicious agent to pose its attack may well be eliminated. We are therefore interested in how the effectiveness of an attack varies with its size.

The performance of any system can only be measured with respect to some utility function which models the usefulness of a recommendation. Clearly, the utility of a recommendation is dependent on the perspective of the parties involved. In collaborative recommendation systems, three distinct parties can be identified. First, there are the end-users, to whom the recommendations are delivered. From this perspective, the utility is dependent on how well the recommendation system follows the end-users' personal tastes. Second, there is the database owner, who is primarily interested in the throughput of the system as measured by the total number of transactions. From this perspective, the recommendation system need

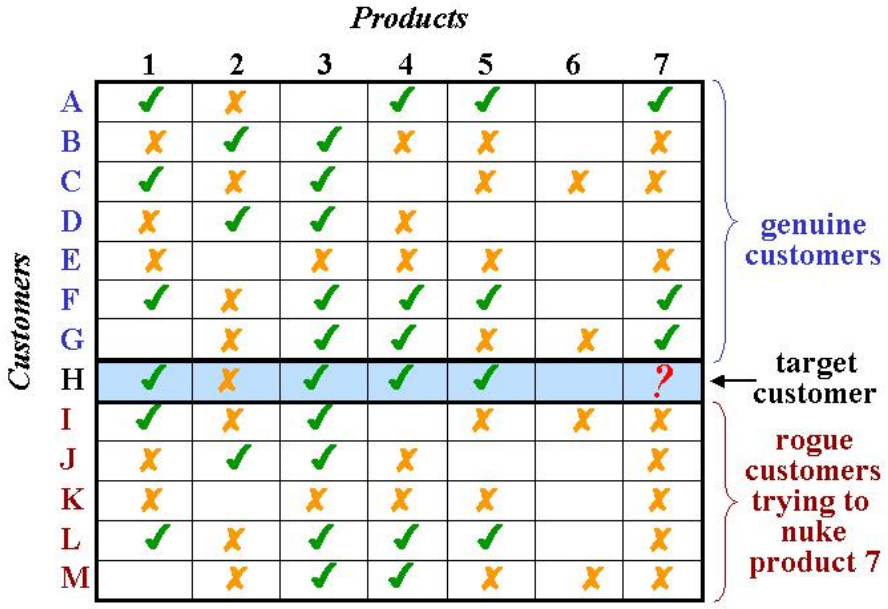

Fig. 1. A simple example of collaborative filtering, and an malicious attack designed to nuke one particular product. Check-marks indicate a positive preference for products and crosses indicate a negative preference.

not actually be accurate, so long as it attracts customers to the service. Third, there are external interested parties who have an indirect interest in the transactions. For a book recommendation system, such a party might be the author or publisher of some of the books that are being recommended. Obviously, these third parties have a vested interest in the recommendations that are made for their product, but do not have direct access to the recommendation system. In our analysis, we take the end-user's perspective, and seek systems which are accurate and consistent in the recommendations that they make to end-users. We examine how the system can be compromised by external third-parties.

3. ACCURACY ANALYSIS

Our analysis of accuracy involves two theoretically justified models of how accuracy varies as a function of the size of the attack. While we are not able to model all aspects of the experiments described in Section 4, we believe that these models are important in a general theory of the robustness of collaborative recommendation.

We assume the standard k-Nearest Neighbours (k-NN) algorithm in our accuracy analysis. This algorithm operates by selecting the k most similar users to the target customer, and formulates a prediction by taking a weighted average of the preferences of these users. Other, e.g. model-based, approaches have been tried

but k-NN is reasonably accurate, widely used and easily analysed.

In machine learning terms, an attack corresponds to the addition of noise to the training data. In general, this noise could be associated with either the class (the item being attacked), the attributes (the other items), or both. We focus exclusively on class noise, and defer attribute noise to future work. The assumption of noise-free attributes corresponds to a rather strong assumption about the attacker. We do not assume that the attacker is omniscient (i.e., knows the entire ratings matrix prior to the attack). However, the attack profiles have noise-free attributes only if the attacker knows the probability distribution over the space of ratings for all products. We are therefore modelling an attacker that lies in the middle of the spectrum between ignorance and omniscience.

We are not concerned with malicious class noise as defined by Kearns and Li [1988]. Rather, we model attacks with a relatively benign noise model that we call *biased class noise*. This model is characterised by two parameters: the noise rate β, and the class bias μ. Noise is added according to the following process. First, an instance is generated according to the underlying distribution. With probability $1-\beta$, the instance is noise-free and labelled by the target concept. Otherwise, with probability $\beta\mu$ the instance is labelled 1 and with probability $\beta(1-\mu)$ the instance is labelled 0. If a sample has been corrupted in this fashion we say that it has (β, μ) biased class noise. Note that biased class noise is not a special case of the much-studied class noise model, in which the correct class label is inverted for some fixed fraction of the training data.

The biased class noise model is natural for representing a variety of stereotypical attacks. For example, a book's author could try to force recommendations of his book by pretending to be numerous customers who all happen to like the book. We call this a PUSH attack and it corresponds to $\mu = 1$. Alternatively, the author's arch-enemy could insert fake customer profiles that all dislike the book; this NUKE attack is modeled with $\mu = 0$.

While our analysis is motivated by collaborative recommendation, our results are relevant to a wide variety of instance-based learning scenarios. In particular, our biased class noise model is deliberately formalized in as generic a manner as possible. For example, biased class noise can model a communication channel that normally transmits a clean signal, but occasionally (a fraction β of the time) distorts the signal (sends 1 a fraction μ of the time and 0 a fraction $1-\mu$ of the time, regardless of the original message).

Before proceeding, we need some additional notation. We assume a $(d+1)$-dimensional instance space $X^d \times \{0,1\}$. In the context of recommendation, instances correspond to users, each dimension corresponds to one of $d+1$ items (one target item whose rating is being predicted, and d other items.) The value on some particular dimension is the user's rating of the corresponding item. The set X captures the permissible item rating values. We allow continuous ratings; without loss of generality we assume $X = [0,1]$. We assume a function $\mathsf{dist}(\cdot,\cdot)$ defined over $X^d \times X^d$, but our analysis does not depend on any particular distance metric.

The set $\{0,1\}$ in the definition of the instance space captures the permissible predicted ratings for some particular target item (i.e., 1 means that the recommender system should recommend the target item to the user). We assume that, for the

purposes of prediction, users either definitely do like a particular item, or definitely do not. However, even though we assume binary predictions, our analysis is entirely compatible with the requirement that actual recommender systems must produce ranked lists of recommended items, rather than simply make binary recommendations. For example, k-NN (like most classification algorithms) computes both a predicted rating as well as a confidence in its prediction based on the neighbour's voting ratio.

Finally, let H be a hypothesis, C be a concept, and D be a probability distribution over X^d. The error rate of H with respect to C and D is defined as $\mathsf{err}(H, C, D) = \Pr_{x \in D}(H(x) \neq C(x))$.

3.1 Absolute accuracy

The first model extends Albert and Aha's noise-free PAC [Valiant 1984; Haussler 1990] results for k-NN [Albert and Aha 1991] to handle biased class noise. We first review these noise-free results, and then state our model as Theorem 3.1.

The key idea behind Albert and Aha's (hereafter: AA) analysis is that of a *sufficiently dense* sample from the instance space. Informally, a subset $S \subset X^d$ is dense if most of the points in the entire space X^d are near many points in the sample S. The terms *most*, *near* and *many* are formalised as follows: Let D be a distribution over X^d. A subset $S \subseteq X^d$ is (k, α, γ)-*dense* if, except for a subset with probability less than γ under D, for every $x \in X^d$, there exists at least k distinct points $x_1, \ldots, x_k \in S$ such that $\mathsf{dist}(x, x_i) \leq \alpha$ for each i.

Given this definition, AA derive [Albert and Aha 1991, Lemma 2.2] a lower bound $\Upsilon_{\text{den}}(k, \alpha, \gamma, |S|)$ on the probability that a sample S of X^d is (k, α, γ)-dense:

$$\Upsilon_{\text{den}}(k, \alpha, \gamma, |S|) = 1 - m^d \sum_{0 \leq k' < k} \Phi_2(\rho, k', |S|),$$

where

$$\begin{aligned} m &= \left\lceil \sqrt{d}/\alpha \right\rceil, \\ \rho &= \frac{\gamma}{m^d}, \\ \Phi_2(\rho, t, s) &= \binom{t}{s} B\left(\max\left\{s/t, \rho\right\}, s, t\right), \text{ and} \\ B(p, s, t) &= p^s (1-p)^{t-s}. \end{aligned}$$

(See Appendix A for a proof.) In the spirit of the PAC model, $\Upsilon_{\text{den}}(k, \alpha, \gamma, |S|)$ is a worst-case lower bound that does not depend on the distribution D over X^d. Note also that $\Upsilon_{\text{den}}(k, \alpha, \gamma, |S|)$ varies with d, but we do not explicitly indicate this dependency because d is fixed by the learning task.

The final step in AA's analysis is to formalise the intuition that k-NN is accurate to the extent that its training sample is sufficiently dense. To describe their result, we first must constrain both the complexity of the concept being learned and the distribution over the instance space. For any constant B, let $\mathcal{D}_B$ be the class of distributions over X^d such that no point has probability exceeding B. For any constant L, let $\mathcal{C}_L$ be the set of concepts C over X^d such that C is the union of a set of regions bounded by closed hyper-curves of total length less than L.

These constraints mean that AA's results hold only for a specific class of learning tasks, and are not truly distribution-free. However, we will see that in fact the model's predictions for real-world tasks are not very sensitive to the values of L and B.

Let $C \in \mathcal{C}_L$ be a concept to be learned, and $D \in \mathcal{D}_B$ be a distribution over X^d. AA's central result [Albert and Aha 1991, Theorem 3.2] is that the probability that the error of k-NN exceeds ϵ when trained on a sample S is at most $\Upsilon_{\text{den}}(k, \epsilon/4LB, \epsilon/2, |S|)$.[1]

The intuition behind AA's results is that k-NN is accurate when training on a sufficient number of instances. We can extend this intuition to handle biased class noise by requiring that, in addition to the sample containing enough *good* (noise-free) instances, it must also not contain too many *bad* (noisy) instances.

We begin by defining sparse subsets analogously to the definition of dense subsets. Informally, a subset $S \subset X^d$ is sparse if most of the points in the entire space X^d are near few points in the sample S. More precisely: Let D be a distribution over X^d. A subset $S \subseteq X^d$ is (k, α, γ)-*sparse* if, except for a subset with probability less than γ under D, for every x there exists at most k points x_i such that $\mathsf{dist}(x, x_i) \leq \alpha$.

Appendix A proves the following lower bound $\Upsilon_{\text{spa}}(k, \alpha, \gamma, |S|)$ on the probability that a sample S of X^d is (k, α, γ)-sparse:

$$\Upsilon_{\text{spa}}(k, \alpha, \gamma, |S|) \;=\; 1 - m^d \sum_{\langle k', k'' \rangle \in \mathcal{K}} \Phi_3(\rho, k', k'', |S|),$$

where m and ρ are as defined above,

$$\begin{aligned}
\Phi_3(\rho, r, s, t) &= \binom{t}{r}\binom{t-r}{s} T\left(\max\left\{r/t, \rho\right\}, r, s/t, s, t\right), \\
T(p, r, q, s, t) &= p^r q^s (1-p-q)^{t-r-s}, \text{ and} \\
\mathcal{K} &= \left\{\langle k', k'' \rangle | 0 \leq k', k'' \leq |S| \wedge k < k' + k'' \leq |S|\right\}.
\end{aligned}$$

To complete our analysis, we observe that the accuracy of k-NN with training data S is equal to the probability that S contains enough *good* instances and not too many *bad* instances. For k-NN, *enough* and *not too many* mean that most of the instances should have at least $\lceil k/2 \rceil$ good neighbours and at most $\lfloor k/2 \rfloor$ bad neighbours.

Let $S_{\text{real}} \subset S$ be the examples corresponding to genuine users, and $S_{\text{attack}} = S \setminus S_{\text{real}}$ be the examples comprising the attack. Of course, we can not know S_{real} and S_{attack} exactly, but we do know that the expected value of $|S_{\text{real}}|$ is $(1-\beta)|S|$ and the expected value of $|S_{\text{attack}}|$ is $\beta|S|$ (where β is the size of the attack), which is sufficient for our analysis.

Furthermore, some of the noisy instances may in fact be correctly labelled. Let

[1] We have departed from AA in several ways. First, AA use the notation k-$\langle \alpha, \gamma \rangle$-*net*; we refer to *denseness* because our biased class noise analysis involves an analogous notion of sparseness. Second, our proof is somewhat different and therefore our bound on the denseness probability differs slightly from AA's. Most importantly, as is standard in the PAC model, AA introduce an additional confidence parameter δ and solve $\Upsilon_{\text{den}}(k, \epsilon/4LB, \epsilon/2, |S|) > 1 - \delta$ for $|S|$, in order to show that k-NN can PAC-learn efficiently. Since robustness is orthogonal to efficiency, we ignore this part of their analysis.

f be the fraction of the instance space labelled 1, and let μ be the class noise bias. Then we expect that a fraction $f\mu + (1-f)(1-\mu)$ of the noisy instances are in fact correctly labelled. Let $S_{\text{good}} \supseteq S_{\text{real}}$ be the instances that are actually labelled correctly, and $S_{\text{bad}} = S \setminus S_{\text{good}} \subseteq S_{\text{attack}}$ be the instances that are actually labelled incorrectly.

As with S_{real} and S_{attack}, we can not know S_{good} or S_{bad} precisely, but can estimate their sizes. Let $|S_{\text{good}}|$ be a random variable characterizing the size of S_{good}, and similarly for $|S_{\text{bad}}|$. The probability that an arbitrary instance of S is correctly labelled is equal to $(1-\beta)+\beta(f\mu+(1-f)(1-\mu))$. The first term is the probability that the instance is real (i.e., in S_{real}) and therefore correctly labelled; the second term is the probability that the instance is part of an attack (i.e., in S_{attack}) but labelled correctly anyway. If we let $\lambda = (1-\beta) + \beta(f\mu + (1-f)(1-\mu)) = \beta(\mu + f - 2\mu f)$ be the "effective" attack size, then we have that the probability that an arbitrary instance of S is correctly labelled is simply $1-\lambda$.

Let "s_{g}" denote the stochastic event that S_{good} contains exactly s_{g} instances. We can calculate $\Pr[s_{\text{g}}]$ as the probability of s_{g} "good" outcomes in a series of $|S|$ binomial trials, where the probability of a "good" outcome is $1-\lambda$:

$$\Pr[s_{\text{g}}] \;=\; \binom{|S|}{s_{\text{g}}} B(1-\lambda, s_{\text{g}}, |S|),$$

where $B(p,s,t) = p^s(1-p)^{t-s}$ was defined above. In accordance with intuition, the expected size of S_{good} is $(1-\lambda)|S|$, and the expected size of S_{bad} is $\lambda|S|$.

To make accurate recommendations, we must have that both S_{good} is $(\lceil k/2 \rceil, \alpha_1, \gamma_1)$-dense and S_{bad} is $(\lfloor k/2 \rfloor, \alpha_2, \gamma_2)$-sparse, where the α_i and γ_i will be discussed below. Let "GD" denote the event that S_{good} is sufficiently (i.e., with respect to α_1 and γ_1) dense, and let "BS" denote the event that S_{bad} is sufficiently sparse. Using this notation, our goal is to calculate $\Pr[\text{GD, BS}]$. We can do so by marginalising over the possible S_{good} sizes:

$$\Pr[\text{GD}, \text{BS}] \;=\; \sum_{0 \le s_{\text{g}} \le |S|} \Pr[\text{GD}, \text{BS} | s_{\text{g}}] \cdot \Pr[s_{\text{g}}].$$

Since S_{good} and S_{bad} are disjoint, the events of some particular instance belonging to one set and the other are clearly (perfectly and negatively) correlated. On the other hand, given particular sizes $|S_{\text{good}}|$ and $|S_{\text{bad}}| = |S| - |S_{\text{good}}|$, the events GD and BS are independent. To see this, note that S_{good} and S_{bad} are generated by independent samples from the underlying distribution D, and therefore the probability that a set is dense or sparse depends only on its size, not on the denseness or sparseness of other samples generated in the same manner. We can therefore decompose the term $\Pr[\text{GD}, \text{BS} | s_{\text{g}}]$ as follows:

$$\Pr[\text{GD}, \text{BS}] = \sum_{0 \le s_{\text{g}} \le |S|} \Pr[\text{GD} | s_{\text{g}}] \cdot \Pr[\text{BS} | s_{\text{g}}] \cdot \Pr[s_{\text{g}}]$$

We can conclude our analysis by noting that to precisely characterize the events GD and BS, we must specify particular values for the distance thresholds α_1 and α_2 and probability thresholds γ_1 and γ_2. Given such values, we can bound the

conditional probabilities of GD and BS as follows:

$$\begin{aligned}\Pr[\mathrm{GD}|s_{\mathrm{g}}] &\geq \Upsilon_{\mathrm{den}}(\lceil k/2\rceil, \alpha_1, \gamma_1, s_{\mathrm{g}}),\\ \Pr[\mathrm{BS}|s_{\mathrm{g}}] &\geq \Upsilon_{\mathrm{spa}}(\lfloor k/2\rfloor, \alpha_2, \gamma_2, |S| - s_{\mathrm{g}}),\end{aligned}$$

and we can therefore bound

$$\Pr[\mathrm{GD}, \mathrm{BS}] \geq \sum_{0\leq s_{\mathrm{g}}\leq |S|} Z(s_{\mathrm{g}}),$$

where

$$Z(s_{\mathrm{g}}) = \Upsilon_{\mathrm{den}}\left(\lceil k/2\rceil, \alpha_1, \gamma_1, s_{\mathrm{g}}\right)\cdot \Upsilon_{\mathrm{spa}}\left(\lfloor k/2\rfloor, \alpha_2, \gamma_2, |S| - s_{\mathrm{g}}\right)\cdot \binom{|S|}{s_{\mathrm{g}}} B(1-\lambda, s_{\mathrm{g}}, |S|)$$

is a lower bound on the probability that S_{good} is $(\lceil k/2\rceil, \alpha_1, \gamma_1)$-dense and S_{bad} is $(\lfloor k/2\rfloor, \alpha_2, \gamma_2)$-sparse, given that $|S_{\mathrm{good}}| = s_{\mathrm{g}}$ and $|S_{\mathrm{bad}}| = |S| - s_{\mathrm{g}}$.

In Appendix A, we prove the following theorem.

THEOREM 3.1 ABSOLUTE ACCURACY. *The following holds for any ϵ, β, μ, d, k, L, B, $C \in \mathcal{C}_L$ and $D \in \mathcal{D}_B$. Let S be a sample of X^d according to D with (β, μ) biased class noise. Let $\lambda = \beta(\mu + f - 2\mu f)$, where f is the fraction of X^d labelled 1 by C. Then we have that*

$$\Pr\left[\mathsf{err}(k\text{-NN}(S), C, D) < \epsilon\right] \geq \sum_{0\leq s_{\mathrm{g}}\leq |S|} Z(s_{\mathrm{g}}),$$

where

$$\begin{aligned}Z(s_{\mathrm{g}}) = {} & \Upsilon_{\mathrm{den}}\left(\left\lceil \frac{k}{2}\right\rceil, \frac{\epsilon}{4LB}, \frac{\epsilon}{4}, s_{\mathrm{g}}\right)\\ & \cdot \Upsilon_{\mathrm{spa}}\left(\left\lfloor \frac{k}{2}\right\rfloor, \frac{\epsilon}{4LB}, \frac{\epsilon}{4}, |S| - s_{\mathrm{g}}\right)\\ & \cdot \binom{|S|}{s_{\mathrm{g}}} B(1-\lambda, s_{\mathrm{g}}, |S|).\end{aligned}$$

So far, our analysis has assumed that all class noise is due to the attack. We now demonstrate that it is straightforward to extend our analysis to also handle noise that arises "naturally" in the collaborative recommendation process.

We begin by extending the biased class noise model with traditional classification noise. In the classification noise model [Angluin and Laird 1988], the noise is modelled with a parameter η as follows: instances are labelled by the target concept C with probability $1-\eta$, and by $\overline{C}$ with probability η. In the extended biased class noise model, each instance is labelled according to the following process. With probability β, the instance is labelled 1 with probability μ and 0 with probability $1-\mu$. With probability $1-\beta$, the instances are labelled by C with probability η and by $\overline{C}$ with probability $1-\eta$.

As above, let f be the fraction of instances labelled 1 by C. Then on average a fraction $\lambda' = \beta\mu(1-f) + \beta(1-\mu)f + (1-\beta)\eta$ of the training instances will be labelled incorrectly by this process. The first term is the probability that an instance is labelled 1 by the biased class noise but the correct label is 0; the second term is the probability that an instance is labelled 0 by the biased class noise but

the correct label is 1; and the third term is the probability that the instance is labelled by $\overline{C}$ by the classification noise. To extend Theorem 3.1 to handle this extended biased class noise model we simply substitute λ' for λ.

To summarise, Theorem 3.1 yields a worst-case lower bound on the accuracy of k-NN under biased class noise. On the positive side, this bound is *absolute* in the sense that it predicts (a probabilistic bound on) the actual error $\mathsf{err}(k\text{-NN}(S), C, D)$ as a function of the sample size $|S|$, noise rate β, and other parameters. In other words, the term *absolute* draws attention to the fact that this model takes account of the actual position along the learning curve. Unfortunately, like most PAC analyses, its bound is very weak (though still useful in practice; see Section 3.3).

3.2 Approximate relative accuracy

In contrast, the second model does not rely on a worst-case analysis and so makes tighter predictions than the first model. On the other hand, the model is only *approximate* because it makes two assumptions. First, it assumes that the training sample is large enough that the learning curve has flattened out. Second, it assumes that, at this flat part of the learning curve, k-NN achieves perfect accuracy except possibly on the boundary of the target concept. We call this second model *approximate* to draw attention to these assumptions, and *relative* to note specifically that it does not predict error on an absolute scale.

To formalise these assumptions, let S be a training sample drawn from the distribution D over X^d, and let C be the target concept. Let S' be the fraction $1-\beta$ of the instances in S that were (correctly) labelled by C during the biased class noise process. Let D' be the distribution that is proportional to D except that $D'[x] = 0$ for all points x on the boundary between C and $X^d \setminus C$. The assumptions of the second model can be expressed as:

$$\mathsf{err}(k\text{-NN}(S'), C, D') = 0 \tag{1}$$

Given this assumption, we can predict the error of k-NN as follows. To classify an instance x using a training set S, k-NN predicts the majority class of the k instances $x_1, \ldots, x_k \in S$ that are closest to x. To classify x correctly, at least $\lceil k/2 \rceil$ of these k instances must have the correct class.

If we randomly draw from D a point $x \in X^d$, there are two cases: either $C(x) = 1$ (which happens with probability f), or $C(x) = 0$ (which happens with probability $1-f$), where as above f is the probability under D that $C(x) = 1$.

In the first case, we need to have at least $\lceil k/2 \rceil$ successes out of k trials in a Bernoulli process where the probability of success is equal to the probability that a neighbour x_i of x will be labelled 1. We can calculate this probability as $(1-\beta)+\beta\mu$. The first term is the probability that x_i is labelled 1 and $x_i \in S'$; by (1), we know that this probability is $1-\beta$. The second term is the probability that x_i is labelled 1 and $x_i \notin S'$. By the definition of the biased class noise process, we know that this probability is $\beta\mu$.

In the second case, again we need at least $\lceil k/2 \rceil$ successes, but with success probability $(1-\beta) + \beta(1-\mu)$, the probability that a neighbour x_i of x will be labelled 0. The first term is the probability that x_i is labelled 0 and $x_i \in S'$, and by (1) this happens with probability $1-\beta$. The second term is the probability that x_i is labelled 0 and $x_i \notin S'$, which occurs with probability $\beta(1-\mu)$.

The following theorem follows from this discussion.

THEOREM 3.2 APPROXIMATE RELATIVE ACCURACY. *The following holds for any β, μ, d, k, C and D. Let f be the fraction of X^d labelled 1 by C. Let S be a sample of X^d according to D with (β, μ) biased class noise. Let S' be the instances of S that were labelled by C. Let D' be the distribution that is proportional to D except that it assigns zero probability to points on the boundary of C. If assumption (1) holds, then*

$$\mathsf{err}(k\text{-NN}(S), C, D') \;=$$
$$1 \;-\; f \cdot \sum_{k'=\lceil \frac{k}{2} \rceil}^{k} \binom{k}{k'} B(1-\beta(1-\mu), k', k) \;-\; (1-f) \cdot \sum_{k'=\lceil \frac{k}{2} \rceil}^{k} \binom{k}{k'} B(1-\beta\mu, k', k).$$

In the absence of additional information, we can not conclude anything about $\mathsf{err}(k\text{-NN}(S), C, D)$ (which is what one can measure empirically) from $\mathsf{err}(k\text{-NN}(S), C, D')$ (the model's prediction) or from $\mathsf{err}(k\text{-NN}(S'), C, D') = 0$ (the assumption underlying the model). For example, if D just so happens to assign zero probability to points on C's boundary, then

$$\mathsf{err}(k\text{-NN}(S'), C, D) = \mathsf{err}(k\text{-NN}(S'), C, D')$$

and so in the best case $\mathsf{err}(k\text{-NN}(S), C, D) = 0$. On the other hand, if all of D's mass is on C's boundary then in the worst case $\mathsf{err}(k\text{-NN}(S), C, D) = 1$. Furthermore, it is generally impossible to know whether $\mathsf{err}(k\text{-NN}(S'), C, D') = 0$.

Despite these difficulties, in the next section we will evaluate the model on real-world data by simply assuming $\mathsf{err}(k\text{-NN}(S'), C, D') = 0$ and $D' = D$, and comparing the predicted and observed error.

3.3 Accuracy Evaluation

We evaluated the two models against two real-world learning tasks:

—The MUSHROOM data-set from the UCI repository contains 8124 instances with 23 attributes, with no missing values.

—The PTV collaborative recommendation data for television listing [http://www.changingworlds.com] contains 2344 instances (people) and 8199 attributes (television programs), and only 0.3% of the matrix entries are non-null. We discarded people who rated fewer than 0.05% of the programs, and programs rated by fewer than 0.05% of the people. The resulting 241 people and 570 programs had a sparseness of 15.5%. The original ratings (values from 1–4) were converted into binary attributes ('like'/'dislike').

We used the standard k-NN algorithm with no attribute or vote weighting. Distance was measured using the Euclidean metric (ignoring non-null attributes). All experiments use $k = 10$.

Our experiments use a variation on the standard cross validation approach. We repeat the following process many times. First, we randomly partition the entire set of instances into a *real* set R, a *fake* set F, and a testing set T. To implement the biased class noise model, a *noisy* set N containing $\beta|R|/(1-\beta)$ instances is then randomly drawn from F. The class attributes of the instances in N are then

modified to 1 with probability μ and 0 with probability $1-\mu$. The k-NN learning algorithm is then training on $R \cup N$ (so the noise rate is $|N|/|R \cup N| = \beta$). We measure accuracy as the fraction of correct predictions for the test instances in T. For MUSHROOM, noise is added only to the class attribute defined by the data-set's authors. For PTV the class attribute (i.e., program to attack) is selected randomly.

3.3.0.1 *Absolute accuracy model..* The absolute accuracy model predicts

$$\Pr[\mathsf{err}(k\text{-NN}(S_\beta), C, D) < \epsilon],$$

the probability that the accuracy exceeds $1-\epsilon$, where S_β is a sample with biased class noise rate β. Let the model's predicted absolute accuracy from Theorem 3.1 be $A_{\text{abs}}(\beta)$. Our empirical estimate $\widehat{A}_{\text{abs}}(\beta)$ of this probability is simply the fraction of trials for which ϵ exceeds the error.

Recall that Theorem 3.1 requires a bound L on the perimeter of the target concept, and a bound B on the probability of any instance under the distribution D. Thus our model is not completely general, and furthermore it is difficult to estimate these parameters for a given learning task. However, it is easily shown that for small values of k, $A_{\text{abs}}(\beta)$ does not depend on L and B, and thus we do not need to tune these parameters of our model for each learning task.

Due to the worst-case analysis, typically $A_{\text{abs}}(\beta) \ll 0$—clearly an absurdity. However, as shown in the top parts of Figs. 2 and 3, the predictions are highly correlated with the empirical values. This figure shows a scatter-plot of the actual and predicted accuracies[2]. The Pearson correlation of these data is 0.88 for PTV and 0.99 for MUSHROOM. The data correspond to $0 \leq \beta \leq 0.6$. The fit is best for small values of β, and the correlation deteriorates for PTV as β approaches unrealistic values close to one.

Figs. 2 and 3 tell us that the predicted absolute accuracies are approximately proportional to their true values. We can therefore use the models to predict robustness, i.e., the sensitivity of accuracy on β. Specifically, we are interested in the increase in error at noise rate β compared to $\beta = 0$. We report results using the ratios $(L - A_{\text{abs}}(\beta))/(L - A_{\text{abs}}(0))$ and $(L - \widehat{A}_{\text{abs}}(\beta))/(L - \widehat{A}_{\text{abs}}(0))$, where $L = A_{\text{abs}}(1)$ is a constant chosen to scale the ratios to [0,1].

The results for MUSHROOM with $\epsilon = 0.25$ are shown in Fig. 4. The predicted and observed accuracies agree reasonably well, even accounting for the fact that the data have been scaled to [0,1]. The fit is by no means perfect, but we are satisfied with these results, since worst-case PAC-like analyses are usually so weak as to be incomparable to real data.

Fig. 5 shows the results for PTV with $\epsilon = 0.3$. Here the fit is worse: PTV appears to be much more robust in practice than predicted, particular as β increases. We conjecture that this is due to the fact that the PTV data is highly noisy, but further analysis is needed to explain these data.

[2]Due to numerical roundoff issues, Figures 2 and 3 show $A_{\text{abs}}(\beta)/m^{2d}$ rather than $A_{\text{abs}}(\beta)$ for the predicated absolute accuracy. Since m and d are constant for a particular data-set, the displayed predictions are proportional to the true predictions.

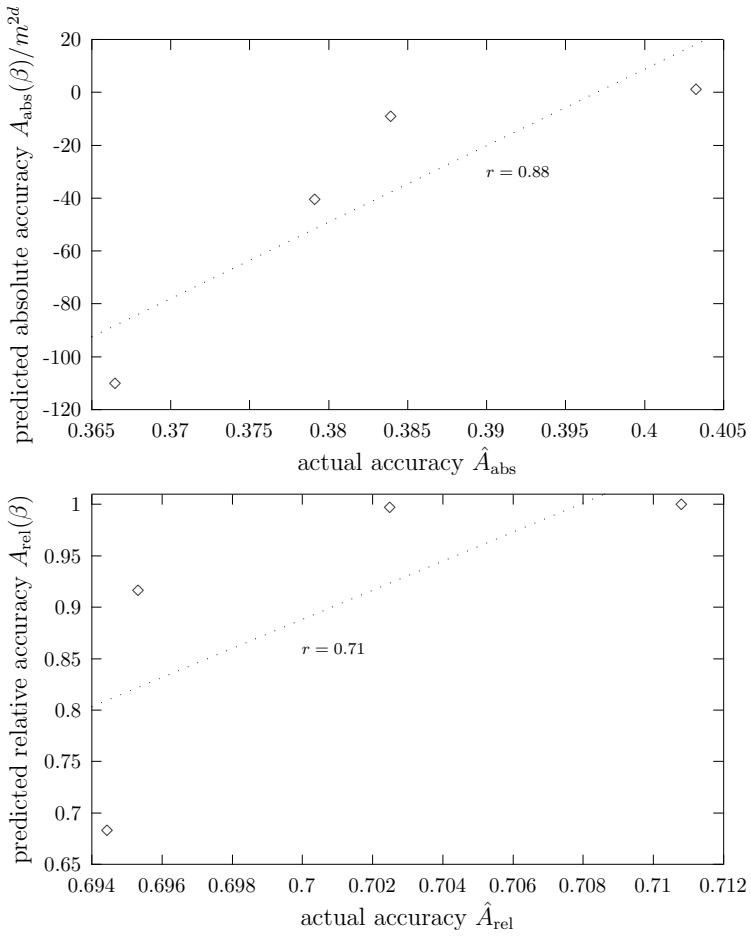

Fig. 2. PTV: Correlation between the actual recommendation accuracies and the bounds predicted by both the absolute (top) and relative (bottom) models.

3.3.0.2 *Relative accuracy model..* The relative accuracy model predicts the error $\mathsf{err}(k\text{-NN}(S_\beta), C, D)$, where as before S_β is a sample with biased class noise rate β. Let $A_{\text{rel}}(\beta)$ be the model's prediction from Theorem 3.2. Our empirical estimate $\widehat{A}_{\text{rel}}(\beta)$ of this probability is simply the fraction of incorrectly classified test instances. As before, we scale all data to [0-1].

The bottom parts of Figs. 2 and 3 shows the correlation between the predicated and actual relative accuracies. For MUSHROOM the Pearson correlation is $r = 0.99$, while $r = 0.71$ for PTV. As with the absolute accuracy model, the correlation gets weaker as β increases to 1 for PTV, but is quite strong over smaller, more reasonable, values.

The results for MUSHROOM are shown in Fig. 6 and the PTV results are shown in Fig. 7. The model fits the observed data quite well in both domains, though as before PTV appears to be inherently noisier than MUSHROOM.

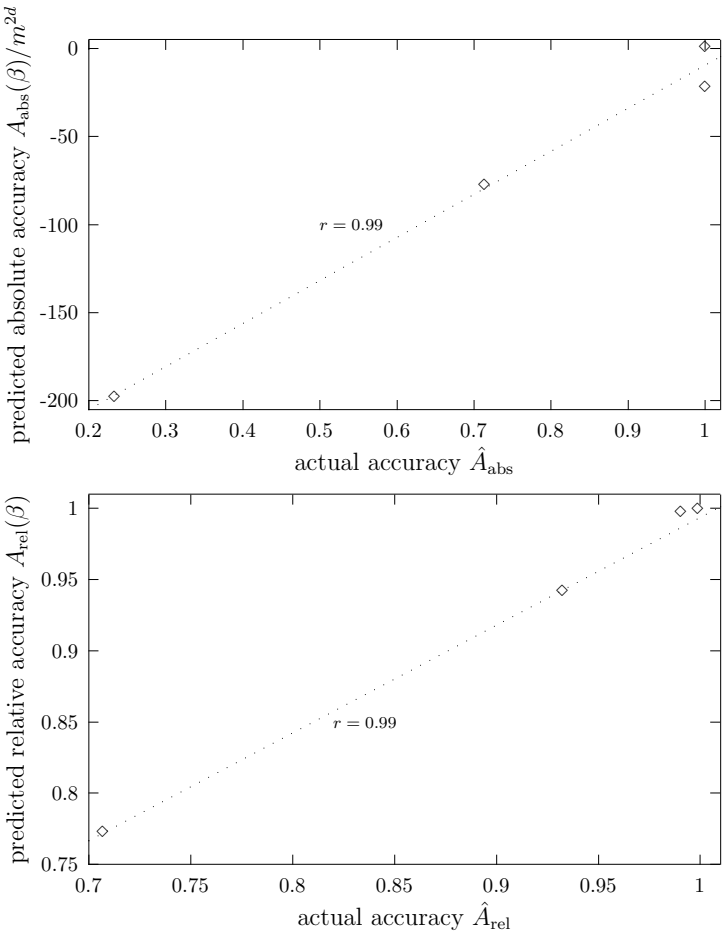

Fig. 3. MUSHROOM: Correlation between the actual recommendation accuracies and the bounds predicted by both the absolute (top) and relative (bottom) models.

3.4 Related work

There has been substantial theoretical analysis and algorithmic work on collaborative recommendation [Kumar et al. 1998; Pennock et al. 2000; Azar et al. 2001; Drineas et al. 2002]. For example, Azar et al. [2001] provide a uniform theoretical analysis for several information retrieval problems, including collaborative recommendation, latent semantic analysis and link-based methods such as hubs/authorities. They cast these problems as matrix reconstruction: given a matrix of objects and their attributes (e.g., for collaborative recommendation, the objects are products, the attributes are customers, and matrix entries store customers' ratings) from which some entries have been deleted, the task is to reconstruct the missing entries (e.g., predict whether a particular customer will like a specific product). Azar et al. [2001] prove that the matrix entries can be efficiently recovered in as long as the original high-rank data can be mapped to some low-rank approximation, which occurs naturally in many settings.

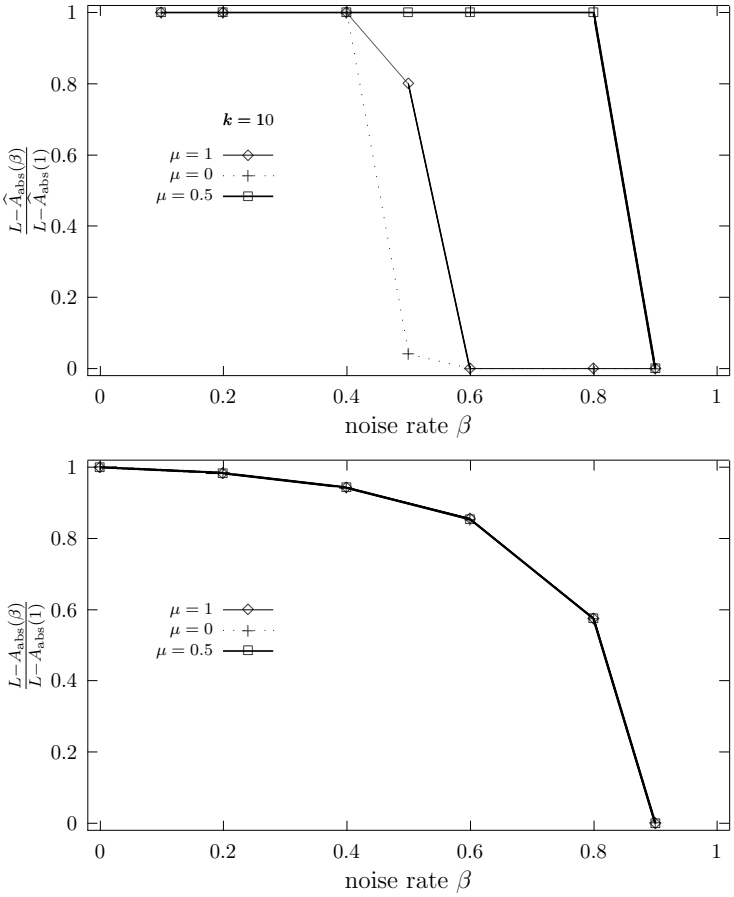

Fig. 4. MUSHROOM: Empirical (top) and predicted (bottom) scaled absolute accuracy.

Given the matrix-reconstruction connection between collaborative recommendation and link-based methods such as hubs/authorities [Kleinberg 1998], it would be interesting to explore the connection between our results and the investigation by Ng et al. [2001] of the sensitivity of the linked-based techniques to perturbations to the document-document link matrix.

The fundamental difference between all of these results and ours is that the biased class noise model is more complicated than simple uniform deletion of the matrix entries. It remains an open question whether these results can be extended to accommodate our model.

4. STABILITY ANALYSIS

In this section, we present an empirical evaluation of robustness from the perspective of stability of prediction. We begin by outlining the framework in which we perform our analysis and, following a review of similar work in the field of Knowledge-Based Systems, we propose a definition of stability of prediction. Fi-

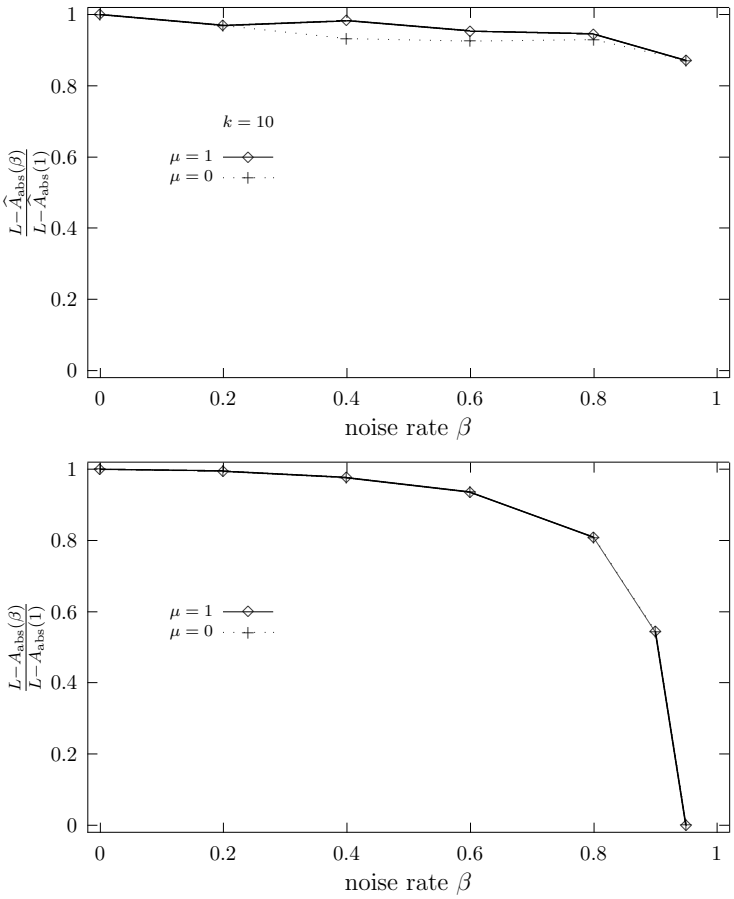

Fig. 5. PTV: Empirical (top) and predicted (bottom) scaled absolute accuracy.

nally, we discuss some attack strategies and present our experimental results.

4.1 Formal Framework

We consider memory-based collaborative filtering (CF) in which the task is to predict the votes of a particular user (the *active* user) from a database of user votes, drawn from a population of other users. Let $\mathcal{U}$ be the universe of all users and fix some set of items I for which users vote. The votes of a user a on all items may be expressed as a vector, termed the *user profile* $\mathbf{v}_a \in V^m$ where V is a discrete set of vote values (including the special symbol $\perp$ interpreted as "not voted on") and m is the total number of items. A CF database D_U for $U \subset \mathcal{U}$ is a collection of votes $\{\mathbf{v}_a | a \in U\}$ for a particular set of users U.

Let $v_{i,j}$ be the j^{th} element of $\mathbf{v}_i$ corresponding to the vote for user i on item j. Using the notation of Breese et al. [1998], define I_i as the set of items on which

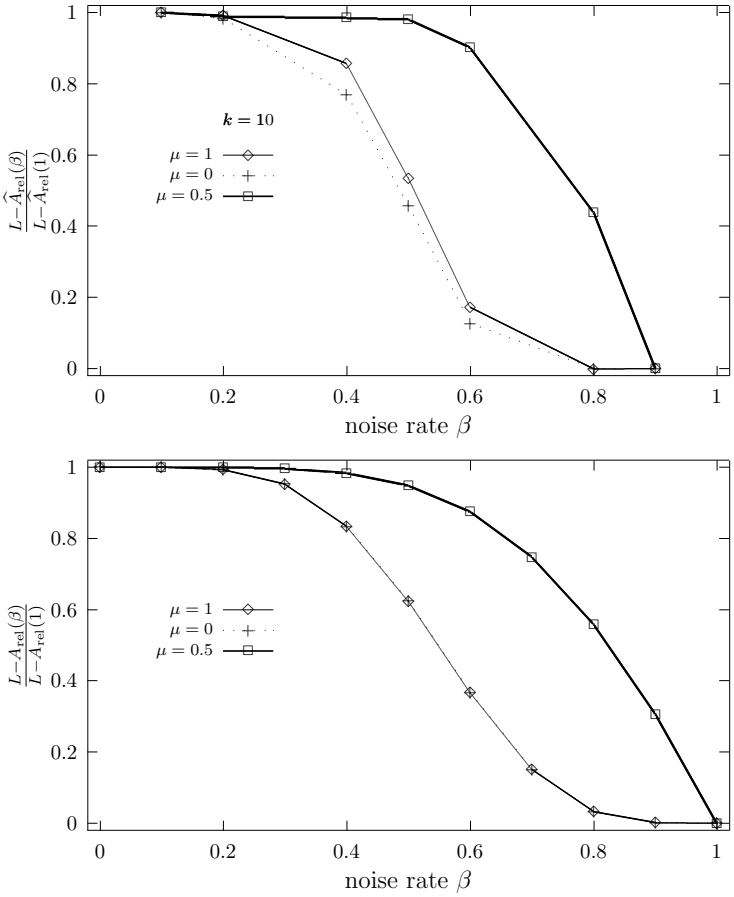

Fig. 6. MUSHROOM: Empirical (top) and predicted (bottom) scaled relative accuracy.

user i has voted. The predicted vote of the active user a for item j, $p_{a,j}$ is given by

$$p_{a,j} = \bar{v}_a + \kappa \sum_{i=1}^{n} w(a,i)(v_{i,j} - \bar{v}_i) \tag{2}$$

where $\bar{v}_i$ is the mean vote for user i, n is the number of users in the database with non-zero weights $w(a,i)$ and κ is a normalising factor.

This equation calculates the average deviation of a neighbour's vote from that neighbour's mean vote, where the mean vote is taken over all items for which the neighbour has voted. The motivation behind this approach is that users' voting distributions may be centered around different points. For example, on a scale of 1 to 5, some users may tend to give relatively low votes for most items, with even liked items only receiving a vote of, say, 3. In contrast, other users may generally give higher votes to items, e.g. 3's, 4's and 5's. Thus it is important to remove such voting bias from the prediction calculation.

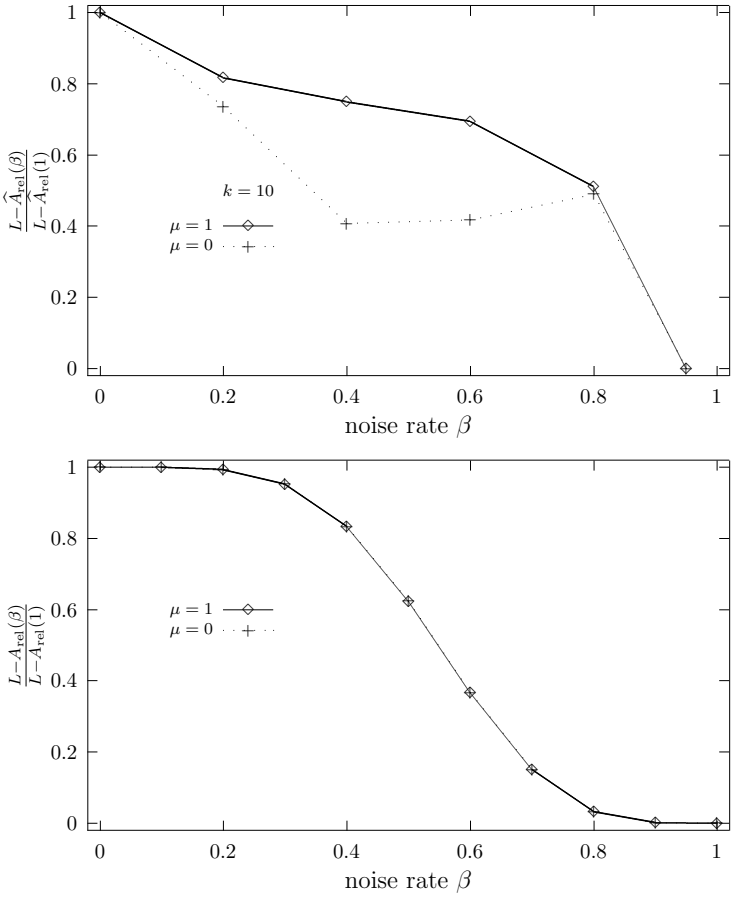

Fig. 7. PTV: Empirical (top) and predicted (bottom) scaled relative accuracy.

4.1.1 *Similarity Measures.* We adopt a nearest neighbour approach and limit the size of the neighbourhood to some value, k. Neighbours are chosen on the basis of greatest similarity $w(a, i)$ to the active user. In this analysis, we consider some of the most widely used similarity measures, namely the Pearson correlation coefficient weighting and the cosine similarity measure. A third measure - Spearman Rank correlation - has also been tested and was found to give similar results to the Pearson correlation measure.

The Pearson correlation coefficient weighting, proposed in Resnick et al. [1994], is defined as

$$w(a,i) = \frac{\sum_j (v_{a,j} - \bar{v}_a)(v_{i,j} - \bar{v}_j)}{\sqrt{\sum_j (v_{a,j} - \bar{v}_a)^2 \sum_j (v_{i,j} - \bar{v}_i)^2}} \tag{3}$$

where the sum is over those items which have been voted for by *both* user a and user i.

The cosine similarity measure, [Breese et al. 1998], is defined as follows

$$w(a,i) = \sum_j \frac{v_{a,j}}{\sqrt{\sum_{k \in I_a} v_{a,k}^2}} \frac{v_{i,j}}{\sqrt{\sum_{k \in I_a} v_{i,k}^2}} \,. \tag{4}$$

A number of enhancements to the standard model have been proposed from a predictive accuracy point-of-view. In this paper, we consider two such enhancements – *significance weighting* and *case amplification* – and investigate the effects of each on stability of prediction.

4.1.2 *Significance Weighting.* Similarity weights calculated on the basis of a small number of common items provide an unreliable measure of "true" similarity between users. Indeed, if the number of common items between users is exactly 2, Pearson and Spearman rank similarity measures always give a result of $+1$, -1 or 0. Therefore, the probability of such users forming a substantial part of a neighbourhood is high, even though they may be, in reality, poor predictors for the active user. Since sparseness is inherent in many collaborative filtering datasets, it is reasonable to expect that many neighbourhoods are at least partially formed by poor quality neighbours, and thus the quality of recommendations is likely to be poor.

To solve this problem, significance weighting has been proposed in Herlocker et al. [1999]. This technique modifies estimated similarity weights based on the number of common items n between users as follows

$$w'_{a,i} = \begin{cases} w_{a,i} \times \frac{n}{N} & \text{if } n < N \\ w_{a,i} & \text{if } n \geq N \end{cases} \tag{5}$$

where N is an arbitrary constant. With this technique, similarity weights calculated on the basis of smaller numbers of common items are appropriately devalued, while weights calculated over large numbers of common items ($\geq N$) are unaffected. Thus, significance weighting captures the intuition that the greater the degree of overlap between users, the more accurate the result is likely to be.

4.1.3 *Case Amplification.* This technique is designed to emphasise weights that are close to 1 and to reduce the influence of lower weights [Breese et al. 1998]. Weights are transformed by the following equation

$$w'_{a,i} = \begin{cases} w_{a,i}^{\rho} & \text{if } w_{a,i} \geq 0 \\ -(-w_{a,i})^{\rho} & \text{if } w_{a,i} < 0 \end{cases} \tag{6}$$

where ρ is an arbitrary constant.

4.2 Related Work

Before we begin our robustness analysis, it is worthwhile to review related work carried out in the area of Knowledge-Based Systems (KBS). One of the underlying claims of KBS systems is that they are able to deal with incomplete, inaccurate or uncertain data. Few researchers in the field have, however, analysed the extent to which this claim is true. To measure the performance of a system, the database upon which predictions are based is typically sub-divided into test and training sets and the predictive accuracy of the test sets is measured. While this process gives an indication of how well a system performs given the training data, it does

not measure how the the performance of a system evolves as a database changes. Thus, in addition to a measure of system accuracy, a measure of system robustness is also required. In this regard, the previous work of Hsu and Knoblock [1995;1996] and Groot et al. [2000], who propose techniques for examining the robustness of knowledge discovery and knowledge based systems, is of interest.

Groot et al. [2000] use as a starting point the informal definition of robustness found in IEEE [1990]:

Definition 4.1. Robustness (Groot et al. [2000]): The degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions.

They develop the discussion by suggesting that robustness should be analysed through a degradation study in which the quality of the KBS output is measured as the quality of the input is gradually decreased. Note here that robustness depends on the particular quality measure that is used. Groot et al. [2000] assume a set of discrete outputs from the KBS and adopt the *precision* and *recall* evaluation measures. They do not, however, propose a single criterion for making a comparison between the robustness of different systems, but rather suggest a number of robustness definitions. For example, one system is considered to be more robust than another if the output decreases at a slower rate with decreasing input quality.

Hsu and Knoblock [1995;1996] give a more precise definition of robustness. In their approach, the focus is on the consistency of a rule set with a database from which it is derived. They argue, intuitively, that the robustness of a rule, r, in the KBS can be measured as the probability that the database is in a state that is consistent with r. As a more tractable definition of robustness, they propose the following:

Definition 4.2. Robustness (Hsu and Knoblock [1995;1996]): Given a database state d and a rule r that is consistent with d, let t denote the transactions on d that result in new database states inconsistent with r. The robustness of r in the database state d is $\text{Robust}(r|d) = 1 - \Pr(t|d)$.

Thus, the robustness of a rule, given a particular database state, depends on the probability that a transaction that moves it into an inconsistent state will occur. However, in their analysis, Hsu and Knoblock [1995;1996] consider only "normal" transactions and therefore examine only the system's robustness to natural noise in the data. In this paper, it is argued that, for open systems (in which a database is updated through system use), it is necessary to consider also how much effort is required on the behalf of external users to deliberately change the output of a system. Thus, it is not appropriate to measure the probability that the transaction will occur, rather the *cost* of such a transaction should be measured. If a transaction that renders a system inconsistent is inexpensive to perform, then it represents a major threat to the integrity of a system.

4.3 Robustness: Stability of Prediction

We approach robustness from the point-of-view of stability of prediction in the following manner. An *attack* is a transformation T which maps a database D_U to a new database state $D'_{U'}$. Under the transformation, each vote, $v_{i,j}$, in D_U is

mapped to a new vote, $v'_{i,j}$. If either $v_{i,j} = \perp$ or $v'_{i,j} = \perp$, this implies the addition of new votes or the deletion of existing votes, respectively. A transformation may also entail the addition of new users, so that $U \subseteq U'$.

Let $\mathcal{T}$ be the set of all possible attacks. Define a cost function $C : \mathcal{T} \rightarrow \mathcal{R}$. In general the cost of a transformation is application dependent. Various criteria can be used to construct a cost function. If the cost is related to the amount of effort it takes to perform the transformation, then this could be modelled by a monotonically increasing function of the number of user-item pairs which are modified by the transformation. There may also be a real cost, if ratings can only be entered into the system by purchasing the item in question.

Fix a set, A, of unrated user-item pairs in the database, which remain unrated in the transformed database i.e. $A \subseteq \{(a, j) | v_{a,j} = v'_{a,j} = \perp\}$. Stability of prediction is defined over this set, A.

Definition 4.3. For each user-item pair $(a, j) \in A$, the *prediction error*, E_p, of prediction pre- and post-attack T is given by

$$E_p(a, j, T) = p'_{a,j} - p_{a,j} \tag{7}$$

where $p_{a,j}$ and $p'_{a,j}$ are pre- and post-attack predictions respectively.

Definition 4.4. The stability of prediction (SOP) of the set A to an attack T is given by

$$\text{SOP}(A, T, \alpha) = 1 - \frac{1}{|A|} \sum_{a \in A} \kappa_{a,j}(\alpha) \tag{8}$$

where α is an arbitrary prediction shift. When $\alpha \geq 0$, $\kappa_{a,j}(\alpha) = 1$ if $E_p(a, j, T) \geq \alpha$, and 0 otherwise; when $\alpha < 0$, $\kappa_{a,j}(\alpha) = 1$ if $E_p(a, j, T) \leq \alpha$, and 0 otherwise. For any given α, an SOP value of 1 indicates no change in pre- and post-attack predictions (or an unsuccessful attack), while a value of 0 means that all predictions have been changed by at least α. With this definition, a comprehensive picture of system stability can easily be obtained by varying α and examining the trends that emerge. For example, an SOP value of 0.5 at $\alpha = 2$ indicates that 50% of all predictions were shifted by at least +2 units on a given rating scale, while an SOP of 0.3 at $\alpha = -1$ indicates that 70% of all predictions were shifted by at least -1 units.

This particular model of robustness measures the power of a recommender system to deliver stable predictions to users in the presence of an arbitrary amount of inaccurate data in a system. As such, this model is independent of the "true" ratings for the items over which SOP is calculated. This is an important feature of the model, since it abstracts system accuracy away from the concept of stability of prediction. Thus, a direct comparison of the stability of different recommender systems is permitted, without regard to accuracy considerations.

The above definition is concerned with the general stability of a system. However, other metrics may be useful depending on the nature of an attack. Consider the class of targeted attacks where the goal is to force item ratings to a particular target value, R_{target}. Product push and nuke are examples of such targeted attacks. Given the set A of user-item pairs and an item j, let $A_j \subseteq A$ be those members of A who

have not rated j. Let $\mathcal{T}_c$ denote the set of all attacks of cost c from this class . We define *power of attack* (POA) for attack $T \in \mathcal{T}_c$ and item j by

$$\text{POA}(A_j, j, T) = \frac{1}{|A_j|} \sum_{a \in A_j} \Big(|R_{target} - p_{a,j}| - |R_{target} - p'_{a,j}| \Big) \tag{9}$$

We can now write the average POA for all such attacks of cost c over set A_j as

$$\text{POA}(A_j, j, c) = \frac{1}{|\mathcal{T}_c|} \sum_{T \in \mathcal{T}_c} \text{POA}(A_j, j, T) \tag{10}$$

Let I be the set of all items in A. Then the average POA of an attack of cost c on set A is:

$$\text{POA}(A, c) = \frac{1}{|I|} \sum_{j \in I} \text{POA}(A_j, j, c) \tag{11}$$

POA gives a positive value when the direction of prediction shift is towards the target. For example, a POA of +1 indicates that the post-attack predicted ratings are 1 rating unit closer to the target on average than the pre-attack predictions. If the direction of prediction shift is away from the target, then POA gives a negative value and if there is no difference in pre- and post-attack predictions, POA is 0.

In our experimental evaluation (Section 4.5), the cost of an attack is a function of the number of attack profiles added to the database, and therefore of the noise rate β.

4.4 Attack Strategies

Two different kinds of attack are considered – namely *product push/nuke attacks* and *random attacks*. The goal of the former is force the predicted rating of an item or group of items to a specific target rating. For example, in a product nuke attack, the goal is to force all the predicted ratings of targeted items to the minimum rating. In a random attack, the goal is to reduce the overall performance of a system as a whole. Thus, the focus is not on particular users or products, rather it is to target all users and items equally in an attempt to compromise the general integrity of a system. The models of Section 3 assume that false profiles are drawn from the same distribution that generated the genuine users. Here, we allow the flexibility to choose false profiles freely, without regard to the distribution of genuine users in a database. We assume that attackers have no direct access to a system database, and we only consider attacks which are implemented by inserting false profiles through the normal user interface.

Clearly, any false profiles added need to be sufficiently close or similar to the target users if they are to have an influence on predicted ratings. In this regard, the choice of similarity measure is important. For the cosine measure, given the formulation of the equation involved, similarity between false and targeted users is, in general, likely to increase with increasing false profile size, assuming that the number of common items between false profiles and targeted users remains high. In addition, cosine similarity is influenced by whether or not common items between users were "liked" – the higher the ratings given for these items the higher the similarity will be. For Pearson correlation, similarity is calculated only over those

items that both users have in common, and depends on how well users have "agreed" on the items in question rather than on the number of such items. Therefore, to ensure a successful attack regardless of the similarity measure used by a system, the formulation of false profiles will need to take all of the above considerations into account.

From (2), the dependence of a prediction on a database D may be written as follows

$$p_{a,j}(D) = \bar{v}_a(a) + \sigma_{a,j}(D)\,. \tag{12}$$

Since the first term depends entirely on the active user, the deviation term $\sigma_{a,j}(D)$ can only be affected by augmenting the database with false profiles. Therefore, for successful attacks, it is necessary to maximise the magnitude of the deviation term. The contribution to this term by any particular neighbour is a function of the rating given to the item for which a prediction is being sought, the mean rating, as well as the similarity between the neighbour and the targeted user. Thus careful selection of items and ratings that comprise false profiles is required. Assuming that it is intractable for an attacker to create a different false profile for each targeted user, it is also desirable that each false profile have a high probability of being located in the neighbourhood of many targeted users. One simple strategy that reflects all of the above considerations is to build false profiles based on highly rated popular items, with the item being attacked set to a low rating. Such a strategy involving popular items ensures good coverage of a database and in addition, such items tend to be given consistent, predictable (high) ratings. From an attacker's perspective, it is not unreasonable to suggest that the formulation of such false profiles is feasible, given that the strategy relies on global characteristics of a system which are generally known or easy to estimate.

One further consideration arises in the case of product push/nuke attacks. To successfully force predictions to a particular target rating, it is necessary to ensure that the magnitude of the deviation term is maximised in the appropriate direction – i.e. either positively or negatively in the case of product push or nuke attacks respectively. Once more, the choice of similarity measure is important in this regard. Since cosine similarity gives a value between 0 and 1, it is straightforward to ensure that all false profiles contribute in the appropriate direction. However, Pearson correlation gives a value between -1 and 1 and, therefore, care needs to be taken if all false profiles are to correlate in the same way with target users. One solution is to choose items that comprise the false profiles from two "groups" – those that are generally rated higher than average in a database and those that are generally rated lower than average. Although some degree of error must be expected on the part of an attacker in choosing such items, it remains, nevertheless, a feasible proposition.

In the next section, we describe some attack scenarios based on the above observations and we empirically evaluate the stability of prediction of real-world datasets subjected to attack. Note that as the attack strategies employed in product push and nuke attacks are essentially the same, the only difference being in the direction of the attack (i.e. the sign of the deviation term $\sigma_{a,j}(D)$), we therefore focus only on implementing product nuke attacks.

4.5 Stability Evaluation

We used two different datasets in our evaluation:

—The MOVIELENS dataset [http://movielens.umn.edu] consists of 943 users, 1,682 movies and contains 100,000 transactions in total. Movies are rated on a scale of 1 to 5.

—The PTV dataset described earlier, comprising 1,542 users, 8,129 TV programs, and has 58,594 transactions. In these experiments we retain the original rating scale of 1 to 4. Note that this dataset is an order of magnitude more sparse than MOVIELENS.

By experiment, we set the number of nearest neighbours for all attacks to 60 for MOVIELENS and to 15 for PTV. An *all but one* protocol is adopted in which the test set is obtained by removing a single user-item pair from the database. A prediction for this pair is then made using (2) using both cosine and similarity measures. In addition, the effects of significance weighting and case amplification on stability of prediction are discussed. For all attacks, the significance weighting and case amplification parameters N and ρ are set to 50 and 2.5 respectively.

4.5.1 *Product Nuke Attacks.* In our first product nuke attack, we exploit an important weakness that exists in the Pearson correlation formula. The particular weakness in question is that the correlation between users is calculated only over the items that *both* users have rated. Hence, users can correlate strongly even though they may have few items in common. With Pearson's formula, the correlation between users who have *exactly* two items in common is always $+1$, -1 or 0. If the attacker adopts a strategy of building false user profiles consisting of the item to be nuked together with two other carefully selected items, then the probability of a successful attack is high. Ideally, two items about which there is strong consensus in the user population are selected to generate the attack profiles.

Thus, in our first attack, we adopt the strategy of 3 item attack profiles, consisting of the item to be nuked set to the minimum rating together with the two most popular items in the databases. These latter items are given ratings according to the following heuristic: the most popular item is assigned the maximum allowed rating, R_{max}, and the other item is assigned a rating of $R_{max} - 1$. Note that if this rating scheme is reversed with targeted users in the databases, the result will be a product push attack instead of the desired product nuke! The identical attack profile is added a number of times. Since this attack can only be successful against those users in the database that have themselves rated the two most popular items (35% of all users for MOVIELENS and 18% for PTV), we present POA according to (11) over all items, but over only those users who have rated both of the items in question.

Fig. 8 shows the results for both databases. In each of the graphs, we show POA for the Pearson and cosine similarity weights (labelled "pe" and "co" respectively) and we also show the effects of using the significance weighting and case amplification algorithm extensions (labelled "sig" and "amp" on the graphs respectively). For MOVIELENS, the attack is successful using Pearson correlation because false profiles largely correlate in the same direction with targeted users. For example, at $\beta = 0.07$ (corresponding to approximately 65 false profiles added), a POA of

approximately 0.9 is achieved, indicating that all post-attack predictions are 0.9 rating units closer to target than before. Due to the small false profile size, the attack is less successful when using cosine similarity. Given that the minimum profile size of genuine users in the dataset is 20, the false profiles are less likely to dominate the prediction shift. Nevertheless, a POA of 0.67 is achieved at $\beta = 0.07$. With significance weighting, the database is robust regardless of similarity measure used, as the influence on prediction of the small-sized false profiles is almost eliminated. Case amplification has an interesting effect when using Pearson correlation – because the false profiles have ideal correlation with targeted users, even a small proportion of false profiles in a neighbourhood will have a significant impact, and this is reflected in high POA values at low attack strengths. However, this effect would be removed by combining case amplification and significance weighting.

For the PTV database, the attack is more successful when using cosine similarity. Given the sparseness of this dataset, the influence of the small false profiles is not significantly reduced and a POA of 0.87 is achieved at $\beta = 0.01$ (corresponding to approximately 15 false profiles added). The attack is not successful using Pearson correlation. From the results we see a slightly negative POA, indicating that more items were pushed than nuked. In this case, the heuristic of assigning a higher rating to the most popular item in the false profiles breaks down. This is due to the sparseness of the dataset, which is an order of magnitude more sparse than the MOVIELENS dataset. As before, the database is robust with significance weighting employed.

In our second product nuke attack, we adopt the strategy of choosing false profile items that come from two distinct groups in the databases. As outlined above, the first group contains items that are generally rated higher than average in the databases (i.e., *liked* items) and items from the second group are generally rated lower than average (*disliked* items). By giving a high rating to the liked items and a lower rating to the disliked items, we can have high confidence that these false profiles will correlate in the same direction with the majority of the targeted users in the databases. In addition, each false profile contains the item to be nuked, set to the minimum rating. False profile sizes are approximately 40 for the attacks carried out on both the MOVIELENS and PTV databases. Experimental results are calculated over all users and a subset of items.

In Fig. 9, we show POA for both datasets. It is clear from the figure that both databases are robust against attack when the cosine similarity weight is used. Even though the false profile size is 40, there is no guarantee that these items have been rated extensively in the databases. In fact, the number of items in common between genuine and false users is small compared to the average number of items that genuine users have in common. Therefore the databases are robust when the cosine similarity weight is used. The attack is much more successful when Pearson correlation is used. This is because the false profiles have been constructed to correlate well (and in the same direction) with target users. For MOVIELENS, the attack achieves a POA of 1.6 at $\beta = 0.06$, indicating a substantial shift towards target. For PTV, the attack is not as successful, where the likelihood of attack heuristics breaking down is increased due to the very sparse nature of the dataset, resulting in a more modest POA of 0.55 achieved at $\beta = 0.008$. Note that the

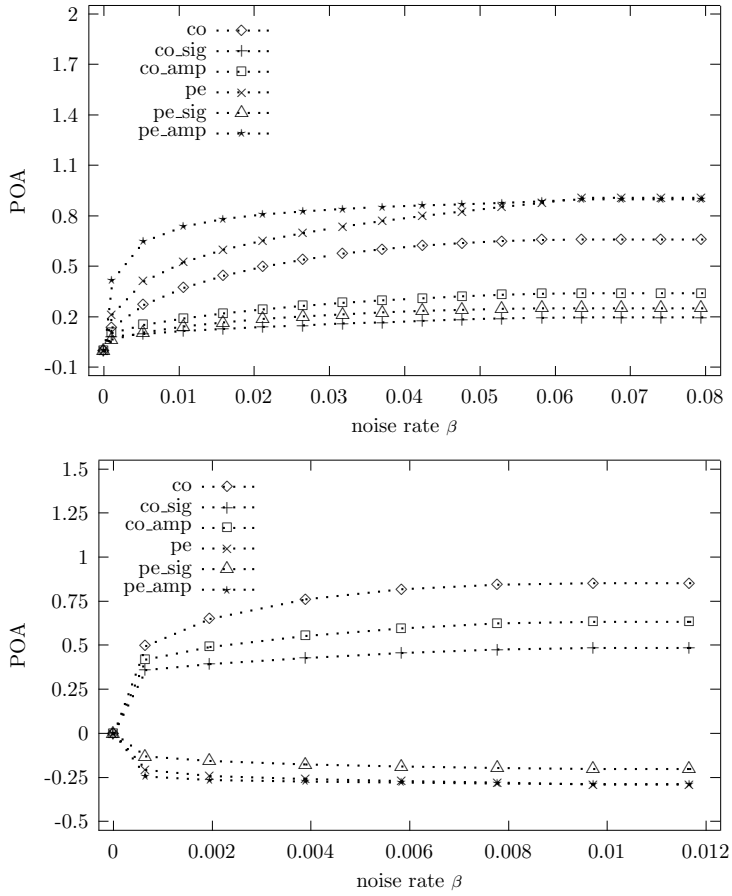

Fig. 8. Product Nuke Attack (1): POA for MOVIELENS (top) and PTV (bottom).

databases are, as before, more robust when significance weighting is incorporated – since the number of common items between genuine and false users is relatively small, the influence of false profiles is reduced.

4.5.2 *Random Attacks.* We have used two different strategies for this kind of attack. In the first, the number of items contained in the false profiles is randomly selected, along with the items and ratings that comprise them. However, we found that both MOVIELENS and PTV were robust to this form of attack. As stated above, false profiles need to be sufficiently close to target users if they are to have an impact. Therefore, since the distribution of the randomly generated profiles is significantly different to the distribution of genuine users in the databases, the false profiles have virtually no effect on predictions.

In the second random attack strategy, the goal is to randomly attack each item in turn in the databases. The false profiles contain the most popular items in the databases, all of which are given a high rating (following the heuristic that popular items are rated highly), together with the item being targeted, which is assigned

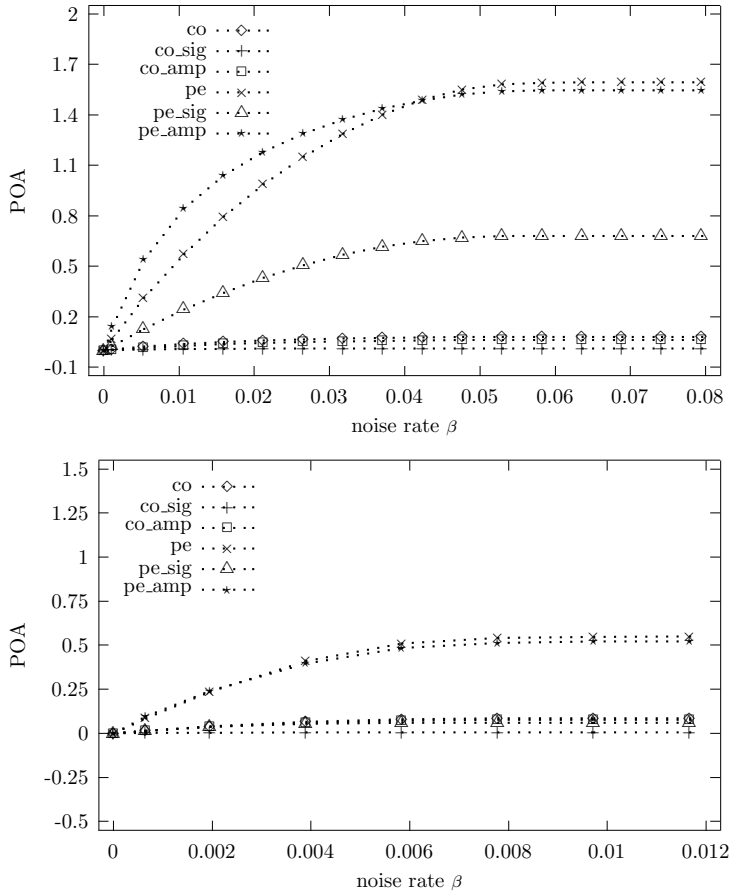

Fig. 9. Product Nuke Attack (2): POA for MOVIELENS (top) and PTV (bottom).

the minimum rating. With this arrangement, we can expect that the false profiles are as likely to correlate negatively as they are positively with targeted users when Pearson correlation is used. In addition, we can expect the number of common items between false and genuine users to be high. When using cosine similarity, however, shifts will be uni-directional, since cosine gives a value between 0 and 1, and given the particular formulation of the false profiles, a product nuke attack will, in effect, be realised. In the experiments, false profile sizes of 60 are used and results are calculated over all users and items.

Figs. 10 and 11 show SOP according to (8) against prediction shift for both databases, using Pearson correlation and cosine similarity respectively. For both databases using Pearson correlation, the attack is very successful, with as many items being pushed as well as nuked. From Fig. 10, we see that, at the strongest attack strength, approximately 50% of all predictions were shifted by at least ± 2 for MOVIELENS, and approximately 60% of all predictions were shifted by at least ± 2 for PTV. These results indicate that we have effectively reduced the performance

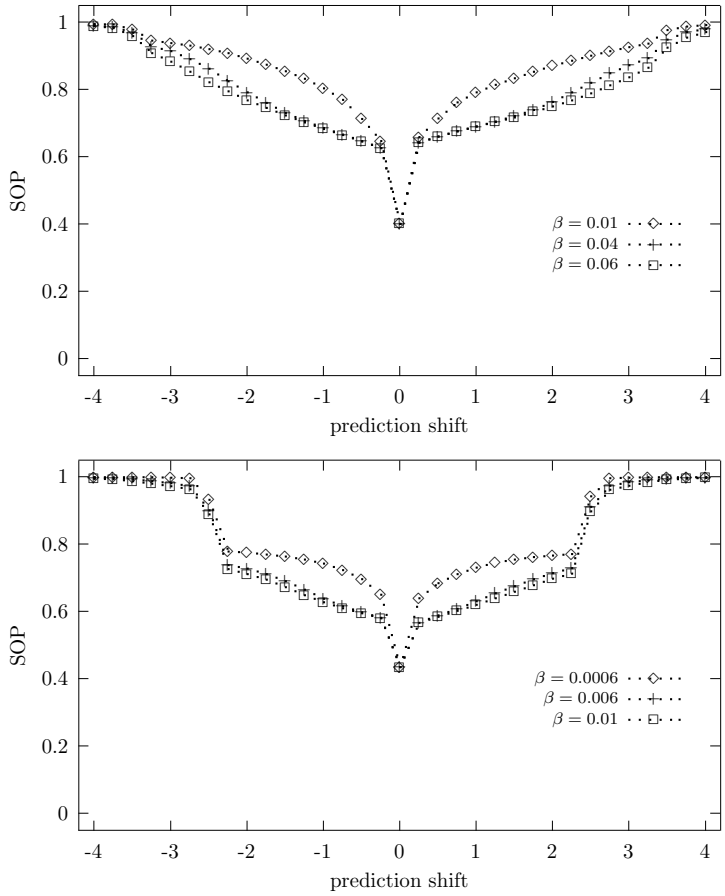

Fig. 10. Random Attack: SOP for MOVIELENS (top) and PTV (bottom) using Pearson Correlation.

of the algorithm to something akin to just reporting the average user rating as a prediction for each user.

When the cosine similarity measure is used, the attack results in, as stated above, a product nuke. Fig. 11 shows that the attack is very successful for both databases, where, at the strongest attack strengths, 53% of all predictions were shifted by at least -2 for MOVIELENS, and 77% of all predictions were similarly shifted for PTV. Note that for both databases, these shifts are sufficient to change predictions from "like" to "dislike". The attack is more successful for PTV – given the sparseness of the database, the large size of the false profiles and the fact that they are comprised of the most popular items in the database gives them a substantial influential on predictions.

A critical point to note is that, given the large size of the false profiles involved, significance weighting does not provide a protection against this attack strategy, regardless of similarity measure used. Thus, this form of attack poses a real threat

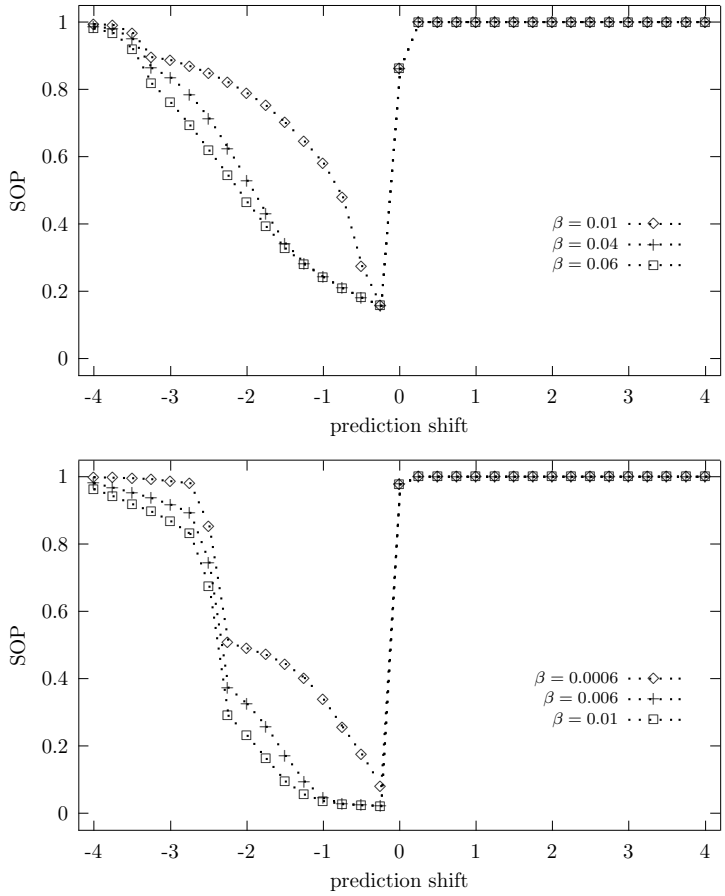

Fig. 11. Random Attack: SOP for MOVIELENS (top) and PTV (bottom) using Cosine Similarity.

to collaborative recommender systems. While we are aware of other extensions to the basic collaborative filtering algorithm that may reduce the effectiveness of this attack (e.g. inverse user frequency [Breese et al. 1998]), nevertheless we feel that our work has shown that recommender systems are indeed susceptible to an attack that is feasible to implement, and that developers ought to be aware of this vulnerability. In future work, we will focus on investigating other forms of attack, and examine how to make systems more robust against attack.

5. DISCUSSION

Collaborative recommendation has been demonstrated empirically, and has been widely adopted commercially. Unfortunately, we do not yet have a general predictive theory for when and why collaborative recommendation is effective. We have contributed to such a theory by analysing robustness, a recommender system's resilience to potentially malicious perturbations in the customer/product rating matrix.

This investigation is both practically relevant for enterprises wondering whether collaborative recommendation leaves their marketing operations open to attack, and theoretically interesting for the light it sheds on a comprehensive theory of collaborative recommendation.

We developed and evaluated two models for predicting the degradation in predictive accuracy as a function of the size of the attack and other problem parameters. The first model uses PAC-theoretic techniques to predict a bound on accuracy. This model is *absolute* in that it takes account of the exact position of the system along the learning curve, but as a worst-case model it is problematic to evaluate its predictions . In contrast, the second model makes tighter predictions, but is *relative* in the sense that it assumes perfect prediction in the absence of the malicious attack. Our preliminary evaluation of the model against two real-world data-sets demonstrates that our model fits the observed data reasonably well.

Regarding prediction stability, we have presented a framework for evaluating the stability of a system under attack. We have outlined several product nuke and random attack strategies, and in our results have shown that collaborative recommender systems are, indeed, vulnerable to attack. In particular, we have highlighted the importance of significance weighting as a defense against certain forms of attack, but have shown it to be ineffective against one of the attack strategies outlined. While we acknowledge that additional modifications to the standard algorithm may impact on the effectiveness of these attacks, nevertheless we feel that the insights provided by our work are valuable.

We are currently extending our work in several ways. First, our models of accuracy assume no attribute noise, which means that they cannot model the attacks that exploit the weakness in the Pearson correlation metric described in Section 4.5.1. We are currently examining whether a general model that accounts for attribute noise is possible. Second, we are developing the same kind of rigorous predictive models for stability as have been described for accuracy. Finally, we are exploring whether the observations reported here can be leveraged to design novel collaborative recommendation algorithms that are more robust than current techniques.

A. PROOF OF THEOREM 3.1

We begin with bounds on the probabilities of certain outcomes in binomial and trinomial processes whose parameters are only partially known.

LEMMA A.1. *Consider a binomial process where outcome o_1 has probability at least ρ and outcome o_2 consumes the remaining probability mass. The probability of exactly s o_1-outcomes in t trials is at most*

$$\Phi_2(\rho, s, t) = \binom{t}{s} B\left(\max\left\{s/t, \rho\right\}, s, t\right),$$

where

$$B(p, s, t) = p^s (1-p)^{t-s}.$$

PROOF. $\binom{t}{s} B(p, s, t)$ is the probability of s successes in t trials of a binomial process with success probability p. We need an upper-bound on this probability

when we know only that $p \geq \rho$. $\frac{\partial B}{\partial p} = 0$ at $p = \frac{s}{t}$. B increases over $0 \leq p \leq \frac{s}{t}$ and decreases over $\frac{s}{t} \leq p \leq 1$. Therefore B is maximal at $p = \frac{s}{t}$, and $\max\left\{\frac{s}{t}, \rho\right\}$ is the value of p that maximises B subject to the constraint that $p \geq \rho$. □

LEMMA A.2. *Consider a trinomial process where outcome o_1 has probability at least ρ and outcomes o_2 and o_3 consume the remaining probability mass. The probability of exactly r o_1-outcomes and s o_2-outcomes in t trials is at most $\Phi_3(\rho, r, s, t) = \binom{t}{r}\binom{t-r}{s} T\left(\max\left\{\frac{r}{t}, \rho\right\}, r, \frac{s}{t}, s, t\right)$, where $T(p, r, q, s, t) = p^r q^s (1-p-q)^{t-r-s}$.*

PROOF. $\binom{t}{r}\binom{t-r}{s} T(p, r, q, s, t)$ is the probability of r o_1-outcomes and s o_2-outcomes in t trials of a trinomial process where o_1 has probability p and o_2 has probability q. We need an upper-bound on this probability when we know only that $p \geq \rho$. $\frac{\partial T}{\partial p} = 0$ at $p = \frac{r}{t}$, and $\frac{\partial T}{\partial q} = 0$ at $q = \frac{s}{t}$. T increases over $0 \leq p \leq \frac{r}{t}$ and $0 \leq q \leq \frac{s}{t}$, and decreases over $\frac{r}{t} \leq p \leq 1$ and $\frac{s}{t} \leq q \leq 1$. Therefore T is maximal at $p = \frac{r}{t}$ and $q = \frac{s}{t}$, and $\max\left\{\frac{r}{t}, \rho\right\}$ is the value of p and $\frac{s}{t}$ is the value of q that maximise T subject to the constraint that $p \geq \rho$. □

The following lemma bounds the probability that a subset is sparse or dense.

LEMMA A.3. *The following holds for any d, α, γ, k, and distribution D. Let $m = \lceil \frac{\sqrt{d}}{\alpha} \rceil$ and $\rho = \frac{\gamma}{m^d}$. The probability that a sample S of X^d drawn according to D is (k, α, γ)-dense is at least*

$$\Upsilon_{\mathrm{d}}(k, \alpha, \gamma, |S|) \;=\; 1 - m^d \sum_{0 \leq k' < k} \Phi_2(\rho, k', |S|), \tag{13}$$

and the probability that S is (k, α, γ)-sparse is at least

$$\Upsilon_{\mathrm{s}}(k, \alpha, \gamma, |S|) \;=\; 1 - m^d \sum_{\langle k', k'' \rangle \in \mathcal{K}} \Phi_3(\rho, k', k'', |S|), \tag{14}$$

where $\mathcal{K} = \{\langle k', k'' \rangle \mid 0 \leq k', k'' \leq |S| \wedge k < k' + k'' \leq |S|\}$.

PROOF. First consider (13). Partition X^d into m^d squares, where m is chosen large enough so that any two points in one square are at most distance α apart. By the Pythagorean Theorem, we require that $m \geq \frac{\sqrt{d}}{\alpha}$, so choose $m = \lceil \frac{\sqrt{d}}{\alpha} \rceil$. Let F be the set of *frequent* squares: those with probability at least $\rho = \frac{\gamma}{m^d}$. Since there are at most m^d squares not in F, the total probability of the non-frequent squares is at most $m^d \rho = m^d \frac{\gamma}{m^d} = \gamma$. If at least k sample points lie in each of the frequent squares, then the sample will be sufficiently dense for the points in the frequent squares. The probability of selecting a point in some particular heavy square is at least ρ, and the probability of not selecting a point in this square is at most $1 - \rho$. By Lemma A.1, the probability of selecting exactly k' out of $|S|$ points in some particular heavy square is at most $\Phi_2(\rho, k', |S|)$. Therefore the probability of selecting fewer than k points in some particular heavy square is at most $\sum_{0 \leq k' < k} \Phi_2(\rho, k', |S|)$. This expression is an upper bound on the probability that the sample is insufficiently dense for some particular frequent square. Since there are at most m^d frequent squares, the probability that the sample is insufficiently dense for one or more frequent squares is at most $m^d \sum_{0 \leq k' < k} \Phi_2(\rho, k', |S|)$. (13) follows by noting that the probability that the sample is sufficiently dense for every frequent square is therefore at least $1 - m^d \sum_{0 \leq k' < k} \Phi_2(\rho, k', |S|)$.

We now prove (14). As before, partition X^d into m^d squares. The *border* of a square is the set of points outside but within α of the square. If at most k sample points lie in each of the frequent squares and their borders, then the sample will be sufficiently sparse for the points in the frequent squares. Consider some frequent square in the set F of squares with probability at least $\rho = \frac{\gamma}{m^d}$. By Lemma A.2, we have that $\Phi_3(\rho, k', k'', |S|)$ is an upper bound on the probability that k' sample points fall in some specific heavy square and k'' points fall in its border. Therefore the probability of selecting more than k points in some particular frequent square or its border is at most $\sum_{k',k''} \Phi_3(\rho, k', k'', |S|)$, where the sum is over the combinations of k' and k'' satisfying $0 \leq k', k'' \leq |S|$ and $k < k' + k'' \leq |S|$. Since there are at most m^d frequent points, (14) follows by noting that the probability that the sample is sufficiently dense for every frequent point is at least one minus m^d times this expression. □

Finally, Theorem 3.1 follows from minor extensions to the proof of AA's Theorem 3.2 described in Albert and Aha [1991]. Here we summarize the argument; see Albert and Aha [1991] for complete details.

Let $C \in \mathcal{C}_L$ be the target concept. Let $C_\alpha^\subset \subset C$ be the *core* of the concept: the points at least a distance α away from some point not in C. Let $C_\alpha^\supset \supset C$ be the concept's *neighbourhood*: the points within a distance α of some point in C. As introduced in Section 3.1, let S_{good} be the noise-free instances in a sample S and $S_{\text{bad}} = S \setminus S_{\text{good}}$ be the noisy instances. Suppose that S_{good} is $\langle \lceil k/2 \rceil, \alpha, \gamma \rangle$-dense and S_{bad} is $\langle \lfloor k/2 \rfloor, \alpha, \gamma \rangle$-sparse. Let k-NN(S) be the set of points labelled positively by k-NN given training data S. Then, except for a set U of instances with probability at most 2γ, k-NN(S) is bounded within a distance α of the target concept: $(C_\alpha^\subset \setminus U) \subseteq (k\text{-NN}(S) \setminus U) \subseteq (C_\alpha^\supset \setminus U)$.

We can therefore bound the error of k-NN(S) by bounding the probability of $C_\alpha^\supset \setminus C_\alpha^\subset$. The perimeter of C is at most L, and therefore the volume of $C_\alpha^\supset \setminus C_\alpha^\subset$ is at most $2\alpha L$. No point in X^d has probability greater than B, and therefore the probability of $C_\alpha^\supset \setminus C_\alpha^\subset$ is at most $2\alpha LB$.

Therefore, assuming that S_{good} is $\langle \lceil k/2 \rceil, \alpha, \gamma \rangle$-dense and S_{bad} is $\langle \lfloor k/2 \rfloor, \alpha, \gamma \rangle$-sparse, we have that $\mathsf{err}(k\text{-NN}(S), C, D) < 2\gamma + 2\alpha LB$. The first term counts instances whose noisy neighbours outvoted noise-free neighbours, and the second term counts mistakes that might occur on the boundary of C.

To complete the proof, we must ensure that $2\gamma + 2\alpha LB < \epsilon$. Since we seek a lower bound on the probability that $\mathsf{err}(k\text{-NN}(S), C, D) < \epsilon$, we can simply split the total permissible error equally between the two causes: $2\gamma = \epsilon/2$ and $2\alpha LB = \epsilon/2$, or $\gamma = \epsilon/4$ and $\alpha = \epsilon/4LB$. □

ACKNOWLEDGMENT

The authors wish to thank Barry Smyth for the PTV data, Mairtín Keane for helpful discussion, and the Weka developers. Kushmerick was supported by grant N00014-00-1-0021 from the US Office of Naval Research, and grant SFI/01/F.1/C015 from Science Foundation Ireland.

REFERENCES

Albert, M. and Aha, D. 1991. Analyses of instance-based learning algorithms. In *Proc. 9th Nat. Conf. Artificial Intelligence*.

Angluin, D. and Laird, P. 1988. Learning from noisy examples. *Machine Learning 2,* 4, 343–370.

Azar, Y., Fiat, A., Karlin, A., McSherry, F., and Saia, J. 2001. Spectral analysis of data. In *Proc. 32nd ACM Symp. Theory of Computing*.

Breese, J., Heckerman, D., and Kadie, C. 1998. Empirical analysis of predictive algorithms for collaborative filtering. In *Conf. Uncertainty in Artificial Intelligence*.

Drineas, P., Kerenidis, I., and Raghavan, P. 2002. Competetive recommender systems. In *Proc. 32nd ACM Symp. Theory of Computing*.

Goldberg, D., Nichols, D., Oki, B., and Terry, D. 1992. Using collaborative filtering to weave an information tapestry. *C. ACM 35,* 12, 61–70.

Groot, P., van Harmelen, F., and ten Teije, A. 2000. Torture tests: a quantitative analysis for the robustness of knowledge-based systems. In *European Workshop on Knowledge Acquisition, Modelling and Management (EKAW'00)*. LNAI Springer-Verlag.

Haussler, D. 1990. Probably approximately correct learning. In *National Conference on Artificial Intelligence*. 1101–1108.

Herlocker, J., Konstan, J., Borchers, A., and Riedl, J. 1999. An algorithmic framework for performing collaborative filtering. In *Proceedings of ACM SIGIR'99*.

Hsu, C.-N. and Knoblock, C. A. 1995. Estimating the robustness of discovered knowledge. In *Proceedings of the First International Conference on Knwledge Discovery and Data Mining*. Montreal, Canada.

Hsu, C.-N. and Knoblock, C. A. 1996. Discovering robust knowledge from dynamic closed-world data. In *Proceedings of AAAI'96*.

http://movielens.umn.edu.

http://www.changingworlds.com.

IEEE. 1990. *IEEE standard glossary of software engineering terminology*. IEEE Standard 610.12-1990.

Kearns, M. and Li, M. 1988. Learning in the presence of malicious errors. In *Proc. ACM Symp. Theory of Computing*.

Kleinberg, J. 1998. Authoritative sources in a hyperlinked environment. In *Proc. ACM SIAM Symp. Discrete Algorithms*.

Kumar, R., Raghavan, P., Rajagopalan, S., and Tompkins, A. 1998. Recommender systems: A probabilistic analysis. In *Proc. 39th IEEE Symp. Foundations of Computer Science*.

Ng, A., Zheng, A., and Jordan, M. 2001. Link analysis, eigenvectors and stability. In *Proc. 17th Int. Joint Conf. Artificial Intelligence*.

Pennock, D., Horvitz, E., and Giles, L. 2000. Socal choice theory and recommender systems: Analysis of the axiomatic foundations of collabortative filtering. In *National Conference on Artificial Intelligence*. 729–734.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. 1994. An open architecture for collaborative filtering of netnews. In *ACM Conf. Computer Supported Cooperative Work*.

Shardanand, U. and Maes, P. 1994. Social information filtering: Algorithms for automating "word of mouth". In *Proc. Conf. Human Factors in Computing Systems*.

Valiant, L. 1984. A theory of the learnable. *Communications of the ACM*, 1134–1142.

Received ; revised ; accepted