Identifying Attack Models for Secure Recommendation

Robin Burke, Bamshad Mobasher, Roman Zabicki, Runa Bhaumik
School of Computer Science, Telecommunications and Information Systems
DePaul University
Chicago, IL
{rburke, mobasher}@cs.depaul.edu, rfzabick@punkgrok.org, rbhaumik@cs.depaul.edu

ABSTRACT
Publicly-accessible adaptive systems such as recommender systems present a security problem. Attackers, who cannot be readily distinguished from ordinary users, may introduce biased data in an attempt to force the system to "adapt" in a manner advantageous to them. Recent research has begun to examine the vulnerabilities of different recommendation techniques. In this paper, we outline some of the major issues in building secure recommender systems, concentrating in particular on the modeling of attacks.

Keywords
Recommender systems, personalization, security, attack modeling.

INTRODUCTION
Recommendation systems are an increasingly important component of electronic commerce and other information access systems. Users have come to trust personalization and recommendation software to reduce the burden of navigating large information spaces and product catalogs. The preservation of this trust is important both for users and site owners, and is dependent upon the perception of recommender systems as objective, unbiased and accurate. However, because recommendation systems are dependent on external sources of information, such as user profiles, they are vulnerable to attack. If a system generates recommendations collaboratively, that is by user-to-user comparison, hostile users can generate bogus profiles for the purpose of biasing the system's recommendations for or against certain products.

Consider a recommender system that identifies books that users might like to read using a user-based collaborative algorithm. (See [7] for the classic formulation of user-based collaborative filtering.) Alice, having built up a profile from previous visits, returns to the system for new recommendations. Figure 1 shows Alice's profile along with that of seven genuine users. An attacker Eve has inserted profiles (Attack1-5) into the system, all of which give high ratings to her book labeled Item6. Without the attack profiles, the most similar user to Alice would be User3. The prediction associated with Item6 would be 0.0, essentially stating that Item6 is likely to be strongly disliked by the user. If the algorithm used the closest 3 users, the system would still be unlikely to recommend the item.

Eve's attack profiles closely match the profiles of existing users, so when these profiles are in the database, the Attack1 profile is the most similar one to Alice, and would yield a predicted rating of 1.0 for Item6, the opposite of what would have been predicted without the attack. Taking the most similar 3 users in this small database would not offer any defense: Attack1, Attack4 and User3 would be selected and Item6 would still be recommended. So, in this example, the attack is successful, and Alice will get Item6 as a recommendation, regardless of whether this is really the best suggestion for her. She may find the suggestion inappropriate, or worse, she may take the system's advice, buy the book, and then be disappointed by the delivered product.

This paper identifies key issues for the study of secure recommendation, focusing particularly on the identification of attack models. In doing so, we draw particularly on two recent studies. O'Mahony and his colleagues [12] conducted a pioneering study on the problem of the robustness of collaborative recommendation. The authors developed a formal framework for analyzing the vulnerability of kNN-based collaborative filtering, and conducted associated experiments comparing actual systems to their theoretical model. Lam and Riedl [9] have also recently published some empirical studies of attacks against collaborative algorithms.

SECURE RECOMMENDATION

As depicted in Figure 2 we identify six essential components that make up the study of secure recommendation. Attack models, the patterns of interaction that attackers may use to influence the system; algorithms, the methods by which predictions are made; profiling, the techniques by which user profiles are gathered and represented; data sources, the different types of data on which recommendation is based; detection, the process by which attacks against a system can be detected; and response, the actions that can be taken to remove bias injected by an attacker. Not part of the system per se, the topic of evaluation is also important for quantifying the vulnerabilities of systems and comparing different designs.

	Item1	Item2	Item3	Item4	Item5	Item6	Correlation w. Alice
Alice	1.0	0.2	0.6	0.6		?	
User1	0.2		0.8		0.8	0.2	-1
User2	0.2	0.0	0.6		0.2	0.2	0.33
User3	0.8	0.2	0.6	0.6		0.0	0.97
User4	0.6	0.5	0.5		0.5	0.2	0.87
User5		0.6		0.5	0.5	0.2	-1
User6	0.4	0.4		0.5	0.5	0.2	0.0
User7		1.0		0.2	1.0	0.0	-1
Attack1	1.0	0.2	0.6	0.6		1.0	1.0
Attack2	0.2		0.2		0.8	1.0	NaN
Attack3	0.2	0.0	0.6		0.2	1.0	0.33
Attack4	0.8	0.2	0.6	0.6		1.0	0.97
Attack5	0.4	0.4		0.5	0.5	1.0	0.65

Figure 1. A push attack favoring Item6.

Our primary goal in this paper is to provide a framework based on these six dimensions for further research in secure personalization. We briefly discuss each of the components introduced above, however, we focus primarily on identifying and characterizing different attack models.

Data sources

The research cited above has concentrated on recommenders that use collaborative data for making recommendations. Another class of recommender system uses algorithms based not on user ratings alone but also information about the items themselves. A content-based recommender, for example, predicts a particular user's preferences by learning to classify items as liked and disliked based on their features and the user's feedback; see [10] for example. A knowledge-based recommender reasons about the fit between a user's need and the features of available products [2].

These systems are vulnerable to a different form of attack, in which the data sources of content information are attacked. This phenomenon is well-known in the web search industry as "search engine spam"[1]: web page authors attach misleading keywords, metadata and other information to their pages in the hopes of being retrieved in response to popular queries.

While there is little direct defense against falsified content data, combining different sources of data in hybrid systems may offer some defense against attacks against any particular data source. This possibility is discussed further below.

Algorithms

The recent research cited above has concentrated on the characteristics of the kNN algorithm for recommendation as this is the most commonly-used technique. The most typical formulation of the idea is the user-to-user comparison seen in Figure 1.

Item-based collaborative filtering works slightly differently [13]. Instead of comparing users' pattern of ratings of items, the system compares the patterns of preference for each item across all users. [9] examines the application of the kNN technique to item-based collaborative filtering, finding some defensive advantages to this technique for certain attacks.

What the kNN techniques share is that they base their predictions on raw user profile data. A variety of model-based recommendation techniques are also well-studied in the recommender systems literature: Bayesian networks, association rules, decision trees, and latent semantic analysis are a few of the techniques that have been used.

A hybrid recommendation algorithm is one that uses multiple data sources of different types. Much of the success of the Google search engine[2] can be attributed to its use of an authority measure (effectively a collaboratively-derived weight) in addition to standard content-based metrics of similarity in query processing [1]. This hybrid technique means that a page that is misleadingly labeled is much less likely to be retrieved by Google than by a system that uses only content. Google is therefore an example of a hybrid approach to secure recommendation, defending against a biased content attack through the addition of collaborative information.

Hybrid recommendation, combining multiple recommenders of different types, is therefore a promising approach for securing recommender systems. The taxonomy of hybrid recommendation developed in [3] can be a useful guide to the landscape of possible hybrid combinations.

Profiling

The research described in [12] and [9] makes use of explicit ratings data: products are individually and explicitly rated by the user as liked and disliked. Some recommender systems use implicit ratings, ratings that are inferred from user behavior, rather than explicitly provided by the user. (See the research reviewed in [6].) Such data sources have different characteristics than the classic explicit rating scenario. In Web usage mining [4, 15], Web

[1] http://www.google.com/contact/spamreport.html

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

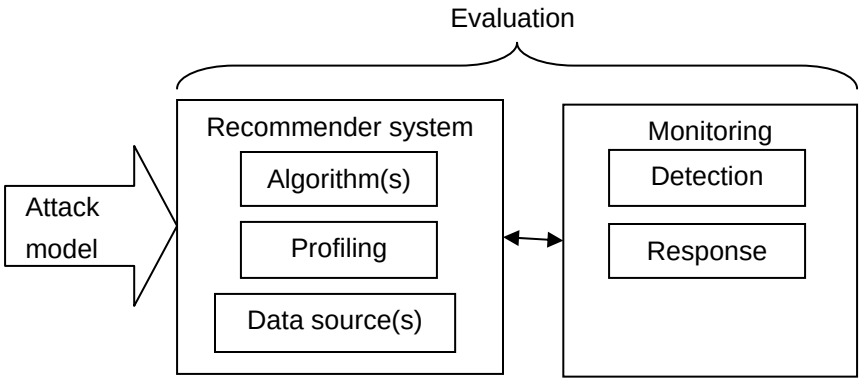

Figure 2. Components of secure recommendation.

server logs are examined for link traversal and dwell time and continuously-valued ratings derived from this analysis, and as a result, negative ratings are not available. We may infer that if page A got less attention than page B that page B is liked more, but not necessarily that page A was disliked. Web usage mining has been studied as the underlying framework for a whole class of Web personalization systems that take users' navigational activity into account to generate recommendations or create adaptive content [11].

Any attempt to secure such recommenders against attack must take into account their unique characteristics both in terms of the structure of the profiles and how profiles are collected. In a web personalization system, an attacker cannot simply call up a list of web pages and assign them ratings. A more sophisticated softbot attack must be mounted in which a software agent interacts with the web site in a predefined way. Such mechanisms are certainly not beyond the capabilities of attackers, and simulations of such attacks will be useful for understanding what distinctive traces they might leave.

Detection and response

Any on-line system may be subject to attack. For most information systems, the ability of an unknown user to change the behavior of the system itself would constitute an attack. For adaptive systems like recommender systems, the whole point is to allow public access and for users' choices to change system behavior. Detection therefore becomes a matter of determining if biased profiles are being injected into the system. Ideally, the most resistant algorithms would be used, the best sources of data and the most effective profiling techniques. However, while attention to these issues will harden the system against attack, they will not eliminate the threat.

Effective detection will depend on the ability to either (i) detect patterns of activity that are characteristic of an attack or (ii) to detect changes in system performance that suggest the presence of biased data. Research into the first option would draw from classic intrusion detection work, in which system behavior is monitored for activity that matches known attack patterns.[3] The second option is intriguing, but perhaps more difficult to realize. [9] demonstrates that an effective attack may not leave any obvious markers, such as a significant degradation of overall accuracy.

The response to an attack will depend on the method by which it is detected. If an attack can be detected by a behavioral measure or the examination of system logs, the profiles generated by the suspect activities can simply be eliminated. However, if bias is detected in a holistic manner, it may be impossible to discriminate the bogus profiles responsible for the bias from those created by genuine users. In this case, the system may need mechanisms to compensate for detected bias without editing the profile base.

Evaluation

There has been considerable research in the area of recommender systems evaluation. See [8] for a particularly comprehensive example. Some of these concepts can also be applied to the evaluation of the security of recommender systems, but in evaluating security, we are interested not in raw performance, but rather in the change in performance induced by an attack.

[12] introduced two evaluation measures: robustness and stability. Robustness measures the performance of the system before and after an attack to determine how the attack affects the system as a whole. Stability looks at the shift in system's ratings for the attacked item induced by the attack profiles.

As Lam and Riedl [9] point out, merely measuring the change in prediction strength may not be sufficient to determine the practical impact of an attack. If an attack changes the predicted rating for an item by some quantity, this may have little or no impact on a user if the item does not appear in the small set of items at the top of the

[3] (Denning, 1987) is a classic example, and there is a large body of continuing research in the area, with the regular RAID symposia serving as a major research forum (http://www.raid-symposium.org/).

recommendation list. Lam and Riedl measure instead how frequently a pushed item is moved into the top recommendation set. This is a more practical measure of the consequences of an attack for end users.

Stability allows us to evaluate how attacking the system benefits the attacker, but only if there is a single outcome of interest. In general, an attacker may have a more complex notion of utility, she may want some products recommended and others not, for example.

To capture a more multi-dimensional notion of the attacker's intent, we can turn to utility theory. The expected utility of the system may be defined as

$$E(u) = \sum_{k=1}^{n} w_k P(k)$$

where k ranges over all possible recommendation outcomes, w_k is the utility associated with that outcome for the attacker, and $P(k)$ is the probability of an outcome. After the system is attacked, we compute $E(u')$ for the biased system, and the denote this change as $\delta = E(u')-E(u)$, the expected utility gain for the attack, or the *payoff*.

Different models of attacker utility can then be employed to the task of evaluating payoff. Adopting a simple notion of utility, we can assign a value to each outcome in which the attacker's favorite item ends up in a user's recommendation list. In this case, δ would correspond to the change in frequency of top list appearance, roughly the measure proposed in [9].

However, some of the attacks described below argue for a more complex notion of utility. In the segmented attack, the attacker is trying to get the system to favor one item against others in a competitive environment. Simply getting the favored item into the top n recommendations might not be acceptable if the other items are also present. Such considerations are best addressed by a utility-based framework for evaluation.

ATTACK MODELING

There has been a number of attack types studied in previous work. This section enumerates these attacks and several additional ones.

Perfect Knowledge Attack

A perfect knowledge attack is one in which the attacker reproduces the precise details of the data distribution within the profile database.

In a perfect knowledge attack, the biased profiles injected by the attacker match exactly with the profiles already in the system except that they exhibit bias for or against some particular item. [12] formalized the perfect knowledge attack and identified its two variants: "push" in which the attacker attempts to have the target item recommended more often and "nuke" in which the aim is to prevent recommendation. We will adopt this terminology with respect to the aims of the attacker. Figure 1 is an example of a perfect knowledge push attack.

Their paper assumes that the bias with respect to class is the only perturbation introduced by the attacker. In other words, push profiles look just like real profiles in terms of their distribution of items and ratings: the only difference is that they give some item a positive rating much more frequently. This assumption lends analytical clarity to the approach, but it is probably not realistic that the attacker will be able to craft an attack so accurately.

Random attack

A random attack may be a push or nuke attack, that is, a particular item will be given high or low ratings, but other ratings in each profile are chosen randomly.

[9] used this attack style, crafting bogus profiles by using a normal distribution based on the overall mean and standard deviation of their dataset. Such profiles would have the same overall characteristics as the dataset, but not the same precise distribution of ratings across individual items.

Average attack

An average attack is a push or nuke attack in which the ratings in crafted profiles are distributed around the mean for each item.

With this attack, we assume that the attacker knows the average rating of each item and the profiles therefore match the distribution across items. Arguing that this level of data was likely to be accessible to an attacker (often exposed by the recommender itself), Lam and Riedl proposed the average attack as more realistic than the perfect knowledge attack, yet still sensitive to the data distribution. They showed that an attack using this level of knowledge was significantly more effective than the random attack.

Consistency attack

A consistency attack is one in which the consistency of the ratings for different items is manipulated rather than their absolute values.

[9] found that an item-based collaborative filtering algorithm was on the whole more robust than the user-based technique against the random and average attacks. This can be seen by looking at Figure 1 from an item-based point of view. An item-based recommender would try to predict the preference of Alice for Item6 by looking at other items that have a similar pattern of preference. Here we see that this particular attack is not successful. In the original, the system would have not had any items with a strong similarity with the entries for Item6. After the attack, the similarity with the pushed item is even worse. Most likely the system would not recommend Item6 in either case, but it would be more likely to recommend it before the attack than after.

What this indicates is that a successful attack against an item-based system would have to have a different character than an attack against a user-based one. The situations are not parallel, because the attacker can add user profiles in their entirety, but only can only augment the item profile.

	Item1	Item2	Item3	Item4	Item5	Item6
Alice	1.0	0.2	0.6	0.6		?
User1	0.2		0.8		0.8	0.2
User2	0.2	0.0	0.6		0.2	0.2
User3	0.8	0.2	0.6	0.6		0.0
User4	0.6	0.5	0.5		0.5	0.2
User5		0.6		0.5	0.5	0.2
User6	0.4	0.4		0.5	0.5	0.2
User7		1.0		0.2	1.0	0.0
Attack1	1.0	1.0	1.0	1.0	0.0	1.0
Attack2	1.0	0.0	1.0	1.0	1.0	1.0
Attack3	1.0	1.0	1.0	1.0	0.0	1.0
Attack4	1.0	0.0	1.0	1.0	1.0	1.0
Attack5	1.0	1.0	1.0	1.0	0.0	1.0
Cosine vs Item6 (pre)	-0.78	0.09	0.14	0.38	-0.70	
Cosine vs Item6 (post)	0.77	-0.09	0.92	0.11	-0.27	

Figure 3. A consistency attack favoring Item6.

To attack an item-based recommender successfully, the attacker must generate profiles for the preferred item that match the profiles of other items that the user likes. For example, Eve needs to make her title match well against items that Alice likes and poorly against others. In other words, Alice's favorite items should get ratings that match the pushed item and other titles should be distant. The specific algorithm used may determine what kind of attack would be successful. Against a median-adjusted algorithm like correlation or some forms of cosine similarity [13], the manipulation of values as in the average attack is not very effective because the bogus profiles essentially add the same thing to their respective items: relatively constant profile values. In order to attack this kind of algorithm the attacker must vary the ratings given to non-favorites across different profiles while using essentially fixed positive values for those favorites that we want to have match against the item to be pushed.

An example of consistency attack can be found in Figure 3. The pattern of ratings in the attacking profiles is the same for Item6 as for Item1, Item3 and Item4, those preferred by Alice. The items not preferred by Alice (Item2 and Item5) get profiles that violate the pattern. As the example shows, the item that Alice likes best Item1 moves from a very low similarity to Item6 to a very high similarity with the attack in place.

In this contrived example, the attacker knows exactly which items Alice likes and dislikes, but in practice, the kind of distributional data used in the average attack can be used to achieve the same effect by selecting items frequently rated high and low. We are currently conducting experiments to quantify the effectiveness of this novel form of attack.

Segmented attack

A segmented attack is an attack against a set of related items – equivalent to a market segment in an e-commerce domain. The aim is to push one item ahead of others considered to be its competitors.

With the segmented attack, the aim of the attacker is not as simple as in the uni-dimensional push or nuke models. In a segmented attack, the attacker's aim is to push a particular item, but also to simultaneously nuke other products within the same segment. The attacker will want to have an effect on recommendations given to precisely those users with an interest in the market segment in question.

For example, Eve may identify other books that share the same general topic as her book Item6 and craft profiles using them. Suppose that, in our example Items 3 and 4 are in the same genre as Item6. An examination of Figure 1 shows that the perfect knowledge push attack does not reduce the recommendation score of Item3 and Item4, and depending on how the score is computed, might even increase it.

Finding the optimal segmented attack is somewhat tricky. A naive approach of combining nuke and push in a single set of attack profiles will not be successful, since such profiles would not be similar to users interested in the genre – all of whom would presumably actually like some of the items that Eve is trying to nuke. Consider what would happen if the attack profiles in Figure 1 had very low scores for Items 3 and 4 – they would lose their similarity to Alice.

Launching a large number of separate nuke attacks, one against each item in the genre and a simultaneous push attack in favor of Item6 is a possible but computationally costly alternative: if Eve is pushing a title with many competitors such as a romance novel, she would have many competing titles to nuke. In addition, such an attack could only guarantee that the competing products would be recommended less often overall and the pushed products more often. It will not guarantee that the pushed item is pushed to Eve's target market.

A successful segmented attack will be one in which the push item is recommended to precisely those users that have many positive ratings in the genre, and in which the presence of similar ratings in the bogus profiles does not

	Item1	Item2	Item3	Item4	Item5	Item6	Correlation w. Alice
Alice	1.0	0.2	0.6	0.6		?	
User1	0.2		0.8		0.8	0.2	-1
User2	0.2	0.0	0.6		0.2	0.2	0.33
User3	0.8	0.2	0.6	0.6		0.0	0.97
User4	0.6	0.5	0.5		0.5	0.2	0.87
User5		0.6		0.5	0.5	0.2	-1
User6	0.4	0.4		0.5	0.5	0.2	0.0
User7		1.0		0.2	1.0	0.0	-1
Attack1	0.9	0.8			0.8	1.0	1.0
Attack2	0.2	0.8			0.8	1.0	-1
Attack3	0.8	0.2			0.8	1.0	1
Attack4	0.9	0.8			0.2	1.0	1
Attack5	0.8	0.2			0.2	1.0	0.65

Figure 4. Bandwagon attack favoring Item6; Items 1, 2 and 5 are the most-commonly

have the effect of increasing the likelihood of the competing products being recommended. Interestingly, it may easier to craft a segmented attack against an item-based algorithm than a user-based one. The goal in the item-based attack would be to add data in which the ratings for the pushed item follow the same pattern as the competing other genre items. We are still investigating the parameters of an optimal segmented attack.

Bandwagon attack

A bandwagon attack is one in which the aim is to associate the pushed item with a fixed set of popular items.

The segmented attack, which aims at particular users interested a particular corner of the item space, is in some ways the opposite of the bandwagon attack. In this attack, the attacker can take advantage of Zipf's law: a small number of items, best-seller books for example, get the lion's share of attention, and hence are more likely to appear in user profiles. By associating her book with current best-sellers, Eve can ensure that her bogus profiles have a good probability of matching any given user, since so many users will have these items on their profiles. The bandwagon attack shown in Figure 4 is even more successful that the perfect attack. Note that it is not necessary to rate the commonly-rated items highly: Alice, for example, does not like Item 2, but most will have some opinion. [12] looked at bandwagon attacks in which ratings for two well-liked items were the basis for push attacks, as in Figure 4. While such attacks could theoretically succeed, a more practical attack would need to target a larger set of popular items since fielded recommender systems will lower bounds on the degree of profile overlap that they will accept.

Probing attack

A probing attack is one in which the aim is to discover the algorithms and/or parameters of the recommender system itself.

As we have shown (and years of experience in computer security confirm), an attacker's knowledge of the system plays a large role in determining the success of an attack. The more Eve knows about the targeted recommender system, the more effective she can make her attack. This is clear with the consistency attack: Eve would only choose such an attack if she knew that the system was item-based.

On one hand, it is a well-established principle of computer security that the fewer secrets the security of a system depends on, the more secure it is [14]. In particular, the security of a recommender system should not depend on its algorithms being unknown – so-called "security through obscurity."

On the other hand, these technical details will not be public knowledge. It may be necessary for an attacker to acquire this knowledge through interaction with the system itself: hence, the probing attack. A probing attack will most likely consist of a series of small-scale attack / test combinations designed to yield enough information to support a later intervention that supports the attacker's real goal.

Given an infinite number of interactions, an attacker would in principle be able to learn everything there is to know about a recommender system. The relevant question for the probing attack is again one of utility: can the marginal benefit of each probe (in terms of increased certainty about algorithm and parameters) be minimized so that such attacks are rendered impractical?

CONCLUSIONS AND FUTURE WORK

Recent research has established the vulnerabilities of the most common collaborative recommendation algorithms. This paper has outlined some of the important issues for continuing research in secure recommender systems: attack models, algorithms, data sources, profiling techniques, detection and response, and evaluation. It is clear that this will be a fruitful area of research for some time.

A recommender system can be considered robust if it can maintain its recommendation quality in the face of attackers' attempts to bias it. The "robustness" property must be predicted on attack type: there is no algorithmic defense against a physical attack that unplugs a server, for example. Attack modeling is therefore a necessary first step to clarifying what we mean by a system's robustness:

against what attacks, on what scale, and with what system knowledge. This paper has outlined four attack models (consistency, segmented and bandwagon) against which our recommendation models have yet to be fully evaluated. Our future work will attempt both to expand this suite of attack models, to evaluate against them a range of different recommender system algorithms and designs, and to explore the problems of detection and response.

ACKNOWLEDGEMENTS

This research is supported in part by the National Science Foundation under the NSF CyberTrust Program, Award #IIS-0430303.

REFERENCES

1. Brin, S. and Page, L. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, *30*, *1–7* (1998), 107–117.
2. Burke, R. Knowledge-based Recommender Systems, in A. Kent (ed.), *Encyclopedia of Library and Information Systems*. Vol. 69, Supplement 32. Marcel Dekker, New York, 2000.
3. Burke, R.. (2002) "Hybrid recommender systems: Survey and experiments". *User Modeling and User-Adapted Interaction*, *12*, *4* (2002),331–370.
4. Cooley, R., Mobasher, B., and Srivastava J. (1999) "Data preparation for mining world wide web browsing patterns". *Journal of Knowledge and Information Systems*, *1*, *1* (1999).
5. Denning, D. E. An Intrusion-Detection Model. *IEEE Trans. on Software Engineering*, *13*, *2* (1987), 222-232.
6. Kelly, D. and Teevan, J. Implicit feedback for inferring user preference: A bibliography. *ACM SIGIR Forum*, *37*, *2* (2003).
7. Herlocker, J. L., Konstan, J. A., Borchers, A., and Riedl, J. T. An Algorithmic Framework for Performing Collaborative Filtering. In *ACM SIGIR 1999*, 230-237.
8. Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. Evaluating Collaborative Filtering Recommender Systems. *ACM Transactions on Information Systems*, *22* , *1* (2004).
9. Lam, S. K. and Riedl, J. Shilling Recommender Systems for Fun and Profit. In *WWW 2004*, New York, May 2004.
10. Lang, K. Newsweeder: Learning to filter news. In *Proc. of the 12th International Conference on Machine Learning*, (Lake Tahoe, CA, 1995), 331-339.
11. Mobasher, B. Web usage mining and personalization, in Singh, M. P. (ed.), *Practical Handbook of Internet Computing*, Chapman Hall & CRC Press, 2004.
12. O'Mahony, M., Silvestre, G. and Hurley, N. Collaborative Recommendation: A Robustness Analysis. *ACM Transactions on Internet Technology*. In press. Accessed at http://www.cs.ucd.ie/staff/nick/-home/research/download/omahony-acmtit2004.pdf.
13. Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. Item-based collaborative filtering recommendation algorithms, in *Proc. of the 10th International WWW Conference* (Hong Kong, May 2001).
14. Schneier, B. Secrecy, Security and Obscurity. *Crypto-Gram*, May 15, 2002. Accessed at http://www.schneier.com/crypto-gram-0205.html.
15. Srivastava, J., Cooley, R., Deshpande, M., and Tan, P. Web usage mining: Discovery and applications of usage patterns from web data. *SIGKDD Explorations*, *1*, *2* (2000), 12–23.