COLLABORATIVE RECOMMENDER SYSTEMS FOR ONLINE SHOPS

Uwe Leimstoll
University of Applied Sciences
Northwestern Switzerland
uwe.leimstoll@fhnw.ch

Henrik Stormer
University of Fribourg, Switzerland
henrik.stormer@unifr.ch

Abstract

Recommender systems are often used in electronic shops in order to suggest similar or related products, potentially interesting products for a given customer or a set of products for a marketing campaign. Most recommender systems use the collaborative filtering method in order to provide the personalization information. The collaborative filtering method is a very efficient and convenient way of achieving personalization as there is no need to introduce semantic information about the products or to manually link products and users together. In the last years, a number of optimizations for collaborative filtering techniques have been developed. This paper collects the ideas and shows which of them could be integrated successfully in order to optimize a collaborative recommender system for online shops.

Keywords

Recommender Systems, Collaborative Filtering, Recommendations, Reference Process, Online Shops

Introduction

In the past years, the number of personalization applications has strongly increased, especially in the field of electronic commerce where personalization becomes an important success factor (Manber et al. 2000, Schubert/Koch 2002). The term personalization means the filtering of information for each particular person in order to provide customers a customized or personalized interaction with a company's products, services, web site and employees (Deitel et al. 2001). The personalization concept is a fundamental requirement for online shops. In contrast to traditional shops, electronic shops cannot provide the personal contact and the individual consultation, which are important means of the customer relationship management.

Recommender systems are one of the most well-known personalization mechanisms. They can be used to suggest similar related or potentially interesting products for a given customer or a set of products for a marketing campaign. Most recommender systems use the collaborative filtering method in order to provide the personalized information. The starting point for collaborative filtering is an m-by-n-matrix (called rating matrix) with m referring to customers (rows) and n

referring to products (columns). By using different techniques, the similarities between the products (item-based technique) or between the users (user-based technique) are calculated.

The information used to fill the rating matrix can either be gained explicitly or implicitly. Explicit information is entered by the customer directly, whereas implicit information is retrieved from the user's interaction with the shop. Explicit information includes product ratings given by the customer. Implicit information includes the orders and the clickstream analysis. The collaborative filtering method is a very efficient and convenient way of achieving personalization, as there is no need to introduce semantic information about the products or to manually link products and users together. The customer's interaction with the shop is the only required information.

When applying recommender systems to online shops, a number of parameters have to be analyzed in order to optimize a recommender system. One parameter is a seasonal product, for example a winter jacket. It is of no use to recommend such a product in summer. The goal of this paper is to show a reference process that can be used when implementing a recommender system for an online shop. The process contains a number of optimization methods, including seasonal products, topic diversification and recommending products when they have already been purchased.

The remainder of the paper is divided into seven sections and is structured as follows: The second section provides a deeper insight into the field of recommender systems. The third and fourth sections show suggestions on some practical aspects of the application and implementation of recommender systems. Based on our experiences, different alternatives of optimizing recommendations are presented in the fifth section. In the sixth section a reference process to facilitate the process of implementing and optimizing a recommender system is proposed. Finally, the seventh section discusses some attributes that could be used to configure the recommender system. The last section contains the conclusion and an outlook.

Recommender Systems Basics

The purpose of recommender systems is to recommend products according to the user's preferences. Well-known applications of recommender systems can be found in the field of books (Linden et al. 2003), music (McCarthy/Anangnost 1998, Chao et al. 2005) and movies (Ling et al. 2005). The broad area of recommender systems has been introduced in the mid-1990s by some early papers on collaborative filtering (Resnick et al. 1994). Meanwhile, the term recommender system is more common because it does comprise content-based filtering, collaborative filtering as well as hybrid approaches.

Recommender System Classification

Recommender systems can be classified in three groups, based on the approach used to generate the recommendations (Adomavicius/Tuzhilin 2005):

1. Content-based filtering approach
2. Collaborative filtering approach
3. Hybrid approach

For the content-based filtering approach attributes are assigned to each product. By using information retrieval techniques on those attributes it is possible to derive the similarity between the products, so that two products with common attributes have a grade of similarity (Basu et al. 1998). The advantage of content-based filtering is the possibility of precisely defining relations between products, namely for cross- or up-selling. However, this advantage comes up at a high price. On the one hand, this approach requires the manual definition of a great number of additional information, e.g. keywords and attributes

for each product. On the other hand, the content-based filtering uses complicated data mining techniques to generate recommendations.

In contrast to content-based filtering, the collaborative filtering approach only needs information about user interaction and transaction such as products ratings, orders or clickstream information in order to provide recommendations. When browsing the websites or buying and rating products, users continuously provide all of this information. Another major difference is that the collaborative filtering approach is based on customer context information. So the strength of this approach is its full automation and its user-based semantic. However, this approach requires a certain amount of data in order to provide valuable results, i.e. the number of customers and, more important, the quantity of users' transactions (often called the cold start problem and the first-rater problem).

The third class of recommender systems uses a hybrid approach, which is a combination of the content-based and the collaborative filtering (Burke 2002). This approach combines the advantages of having a precise description of the relationships between the objects based on the keywords and on the users' interactions. This allows pertinent recommendations from the beginning with a continuous improvement over time by gathering and using more and more users' information.

Another well-known classification distinguishes between the way to pre-calculate results (Adomavicius/Tuzhilin 2005):

1. Memory-based approach
2. Model-based approach

The process for calculating recommendations consists of an offline and online part. In the offline stage, data from the customer's profile is read and processed to improve the performance when calculating recommendations in the online part. Memory-based algorithms do not pre-calculate any results. This is sometimes called lazy learning. Model-based approaches try to create a model in the offline stage. The algorithm presented in this paper falls into this class. Other approaches are based on Bayesian networks (Breese et al. 1998) or statistical techniques (Hofmann 2004).

User-based and Item-based Collaborative Filtering

The collaborative filtering approach can be implemented using user-based or item-based methods. Both take as input the rating matrix with the customers in the row dimension and the products in the column dimension. This two-dimensional matrix represents the relationships between users and products either based on product ratings, purchased products or clickstream data. If product ratings are considered, each element at the intersection of a product and a customer will contain a value between -1 and +1 representing the judgment of the customer for the product where -1 denotes a strong dislike and +1 a strong affection.

When applying the user-based method, in a first step, similarities between users are calculated. This calculation can be achieved by applying different mathematical formulas. In this paper, the similarity between the users is assessed using the cosine method (Resnick et al. 1994). Once the similarities between all users have been calculated, a matrix with the customers on both dimensions and the similarities as entries is returned:

Table 1 Similarities between customers

	Mr. Smith	Mr. Johnson	Mrs. Miller
Mr. Smith		-1	+1
Mr. Johnson	-1		-0.8
Mrs. Miller	+1	-0.8	

Based on this matrix, it is possible for each user to extract the group of most similar users (nearest neighbors). This group is then used in a second step to derive the product recommendations. The principle of the recommendation is pretty obvious; if Mr. Smith is very similar to Mrs. Miller and if Mrs. Miller strongly likes a product that Mr. Smith has not bought yet, the chance that Mr. Smith also likes this product is rather high. The user-based method returns personalized recommendation as each user receives propositions based on his profile. In our example, it is likely that Mr. Smith is fond of the product DVD Lost in Translation because he is very similar to Mrs. Miller who has rated this product high.

In contrast to the user-based method, the item-based method directly derives the similarities between products. Once again, several mathematical approaches can be used to calculate these similarities. In this paper, the methodology of Deshpande and Karypis (2004) has been chosen. This methodology calculates the probability that a product X will be bought if a product Y has already been bought.

A deeper introduction to the common algorithms used for user-based and item-based collaborative filtering as well as an analysis can be found in the paper of Sarwar et al. (2000).

Selecting the Proper Recommender Algorithm and Input Data

Information Source and Matrix Density

When deciding which recommender algorithm to use, the available input data is most important. As stated before, all classes of algorithms offer different advantages and disadvantages. Practically, collaborative filtering algorithms are favored because of their full automation and easy way of implementation.

In order to fill the rating matrix, a collaborative filtering algorithm needs user profile data. In an online shop, the following different information sources can be utilized to fill the rating matrix:

Customer's products ratings: The products ratings are the best source of information an online shop can obtain because they reflect the final judgment of a customer for a given product. Unfortunately, this information is rare, as customers normally rate only the products they have bought and only a rather small group of users does use this functionality. For product ratings, the five star model is very popular. A customer can rate a product with a value between 1 and 5 stars. A linear transformation to the rating matrix would be values of -1 for one star, -0.5 for two stars, 0 for three stars, +0.5 for four stars and +1 for five stars. Other researchers propose to base the ratings on an average rating by a customer (Lemire/Maclachlan 2005).

The bought items derived from the orders: The order information is another good way of deriving the customer's preferences. This information is easily accessible and is much more dense than the product ratings. However, people sometimes buy products they actually do not like. This is the case for a movie they have not seen yet and think they might like it. Therefore an explicit rating is more reliable than order information.

The clickstream information: The clickstream data is the last kind of information, which can lead to the definition of the users taste. This is the most substantial but also the most doubtful information source. There is, however, a correlation between the visited web pages and the user's interest, especially if the user visited several times the same product's page.

External Information: The shop administrator can provide external information for customers. This includes products a customer liked or did not like.

An important factor of this information is the resulting matrix density. Analyses have shown that the quality of the recommendations depends on the matrix density (Sarwar et al. 2000, Stormer et al. 2006). By combining all available information sources, it is possible to increase the matrix density.

Explicit ratings have a much better impact than implicit ones (Herlocker et al. 2004). Therefore, explicit ratings should be used when available.

Product Variants

Important properties of products are variants. With the success of mass customization (Reichwald et al. 2000) variants are becoming more and more popular. A product variant is defined as a distinct unit within a brand or product line that is distinguishable by size, price or appearance (Govers/Schoormans 2005). Typically, it is of no use to distinguish between different variants in a recommender system. Therefore, all variants should be aggregated to one product.

Even more problematic are products that are created individually. Those products are typically not offered in predefined variants but are designed new for each customer. In this case, the matching between similar products becomes problematic as the same product is rarely sold to two different customers.

Implementing an Item-Based Algorithm in Practice

Deshpande and Karypis (2004) describe one of the best algorithms for implementing recommendations using webshop order data. This algorithm expects a binary rating matrix where cell *(i,j)* is equal to 1 if customer *i* has bought product *j*. To weight customers higher that have only bought a few items in contrast to customers that have purchased a large number of different products, this rating matrix is normalized. This is achieved by dividing all customer vectors by the number of products the customer has purchased.

Deshpande and Karypis (2004) define the number of different users that bought a product as *Freq*, thus *Freq(i)* is the number of nonempty entries in the rating matrix for the column of product *i*. The following formula is used to calculate the equality between products (α is a parameter between 0 and 1):

$$sim(p_i, p_j) = \frac{\sum_{\forall q: R_{q,j} > 0} R_{q,j}}{Freq(i) \times (Freq(j))^{\alpha}}$$

When this formula is to be implemented in an online shop, the rating table has to be initialized first. In a simple case, the rating table has three columns, namely productId, userId and rating. By looping through all orders retrieved, this table can easily be filled.

The next step is to calculate the similarity between products using the formula presented above. For this step, each product vector has to be compared with all other vectors. The results could be saved in another item table, containing the columns productId1, productId2 and similarity. For a product x that should be compared to a product y, the numerator of the above presented formula can be calculated by summing up all ratings retrieved by the following SQL statement:

```
SELECT rating FROM rating WHERE productId in
(x,y) GROUP BY userId HAVING COUNT(userId)=2
```

The denominator can be retrieved by two SQL statements, one for the frequency of product x:

```
SELECT count(*) FROM rating WHERE productId=x
```

and the same one with productId=y. All three values need to be combined to get the final similarity. At the end, three SQL SELECT statements and one SQL INSERT statements are sufficient for calculating the similarity between two products. To improve the performance for large rating matrices, caching strategies should be evaluated.

Finally, all recommendations can be read from this table. Two different ways are possible:

Non-personalized recommendations: Non-personalized recommendations for one product with productId x can be extracted from the item table by a simple select statement that returns the products with the highest similarity:

```
SELECT productId2 FROM item WHERE
productId1=x ORDER BY similarity DESC
```

If the database supports to restrict the result set only the first n products can be retrieved.

Personalized recommendation: Personalized recommendations are also possible by using the already bought products of each customer. In a first step, the IDs of all products a customer y has already bought are extracted from the rating table:

```
SELECT productId FROM item WHERE userId=y
```

Then, for all those products the recommendations are retrieved and combined. If a product is contained in more than one recommendation list all similarities are summed up. Fortunately, this can be done with only one SQL statement if (p1, p2, ... , pn) is a list of all productIDs retrieved by the first statement:

```
SELECT productId2 FROM item WHERE productId1 in
(p1,p2,...,pn) GROUP BY productId2
ORDER BY SUM(similarity) desc
```

Optimizations to Improve Recommendations

Time Optimizations

A customer's interest might change over time. Therefore, newer ratings, clickstreams and orders are more important than older ones. Weighting data with a newer timestamp higher can accomplish this. Weng and Liu (2004) suggest to use a self-defined weight greater one and multiply this with the original rating value if the rating was done within the last k days.

Consumer Products

Products typically differ in the number of buys per customer. Products like books or DVDs are typically only sold once, whereas food or detergents are sold more than once to the same customer. Today's recommender systems usually do not recommend items a user has already bought. This strategy could be changed for products that have been sold more than once to the customer. If, for example, a user always buys the same bar of chocolate, the recommender system could recommend this item if his current order does not include the chocolate.

Seasonal Products

A seasonal product is a product that is only sold during a special period within a year. Seasonal products include Christmas trees, winter coats or sunglasses. Figure 1 shows two seasonal products, the working gloves "Wintermercedes" and "88PBWA". The first one, as the name indicates, is designed for the usage in winter times. Therefore, the orders are mainly done during September to February. The glove "88PBWA" is a summer glove that is sold during February to October.

As the example shows, it is not useful to recommend the glove "Wintermercedes" in July because it is typically sold in winter. If an online shop wants to control seasonal products, two steps need to be fulfilled:

1. All seasonal products together with their best selling period need to be identified.
2. The recommender system should recommend a seasonal product only during the selling period.

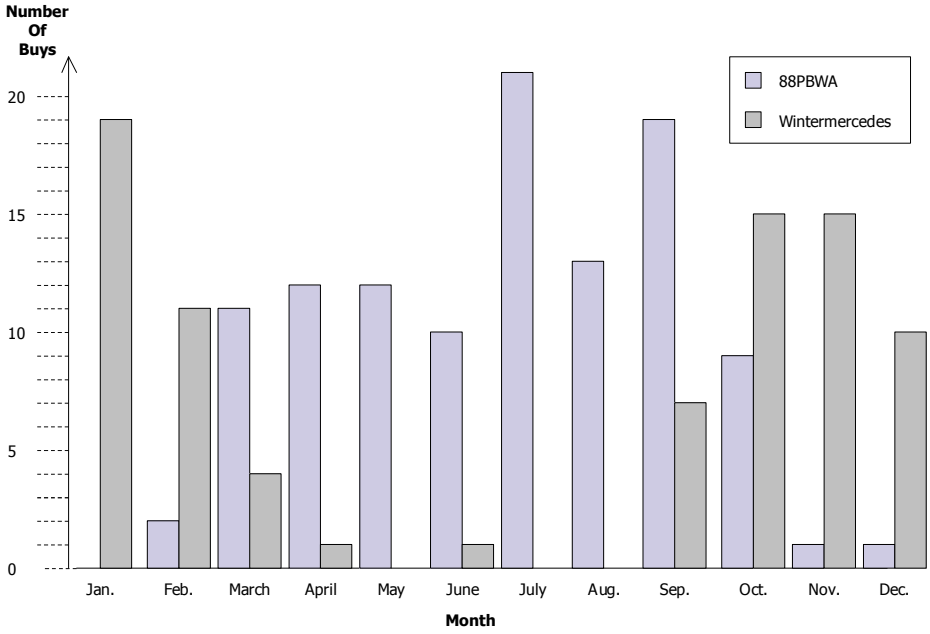

Figure 1 Two seasonal products

The first step can be done automatically by analyzing the ordering information. For the second step, the resulting recommendation list has to be reordered. A general approach is shown in the next paragraph.

Shop Administrator Interests

Typically, a recommender system returns a recommendation list (often called Top-N list) that contains all recommendations for a product or a customer. However, a shop administrator might not be interested in promoting all products within this list. There are a number of reasons why a certain product should or should not be recommended, these include:

Margin: The margin is one of the most important attributes for shop administrators. The margin is the difference between purchase price and selling price. Typically this value differs from product to product. The shop administrator is more interested in selling products with a high margin.

Quality: The quality of the product is another very important attribute. When a low quality product is sold, the chances for service costs are higher compared to a high quality product. Therefore, the shop administrator will try to sell high quality products to avoid service costs. If the online shop offers its customers the opportunity to give explicit product ratings, they can be a good indicator of the product quality. Another good indicator is the experience gained from the service and support of a product, like the number of returns because of quality problems.

Size: Size could be another important attribute. For large products, a big effort has to be made in order to send it to a customer. This includes packing, providing a logistics enterprise that transports the product and finally billing. Additionally, large products occupy a greater amount of the inventory space.

Supplier Charge: In brick and mortar stores, suppliers often pay for putting their products on an emphasized place. The shop administrator could retrieve a supplier charge for recommending their products.

Stock: The amount and kind of items on stock influence the interest of the shop administrator to recommend products. A shop administrator is much more interested in selling items he has on stock than in reordering certain items for individual customers.

In a simple approach, a product can be excluded from the recommendation list, for example by storing a flag that can be turned on manually. More complex approaches use information about the product to reorder the product list.

Recommendation Diversity

The problem of collaborative recommender systems is that the best recommendations are often very similar. An interesting approach to reduce this problem is to optimize the resulting recommendation list by reordering it so that different products are on top. This idea is presented in the work of Ziegler et al. (2005). The algorithm uses an arbitrary taxonomy to calculate the differences between product sets (Ziegler et al. 2004). Afterwards, the recommendation list is reordered to present different products on top according to their taxonomy. Smyth and McClave (2001) have done a similar work. They also try to balance the similarity of a product compared to the other products in the top-n list.

A Reference Process for Implementing and Optimizing a Recommender System

The presented optimizations have to be included in the process of calculating the recommendations. Therefore, this paragraph shows a reference process that includes all presented optimizations. Dependent on the data available and the need to implement the optimizations, the reference process can be implemented also partially by removing the not required steps.

The reference process is divided in two parts. The first part shows the initialization steps needed to calculate the similarities between products. The second part describes the activities that could be accomplished when a top-n product list should be generated to recommend products to the customer.

Initialization process

The initialization is a process that is started to calculate the similarities between the products. Figure 2 shows the initialization process in using a UML activity chart.

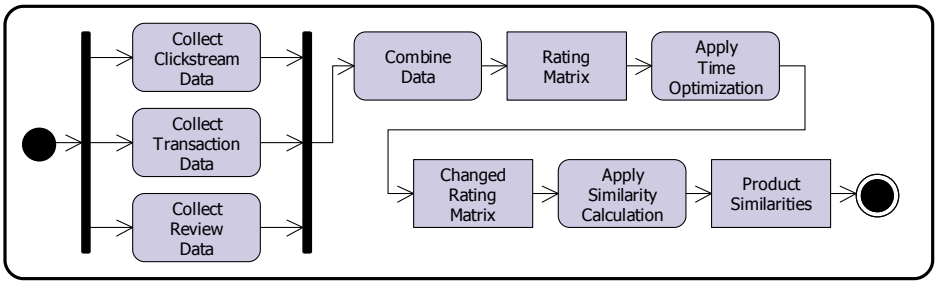

Figure 2 The initialization process

In the first three steps, the available data from the different business systems have to be collected, formatted and evaluated to generate the rating matrix. The combination of the different data available could be done as described. This results in a rating matrix. Then, the above described time optimization could be run. This weights current orders, reviews and clickstreams higher than older ones. The resulting rating matrix is the input for calculating similarities by either implementing an existing algorithm or using an available library. The similarities between the products are typically stored in a database. The initialization process ends with this storage.

Top-N List Generation

The second process describes the necessary activities to retrieve a top-n list for a customer. This process is executed whenever products should be recommended to a customer. The top-n list generation process is presented in Figure 3.

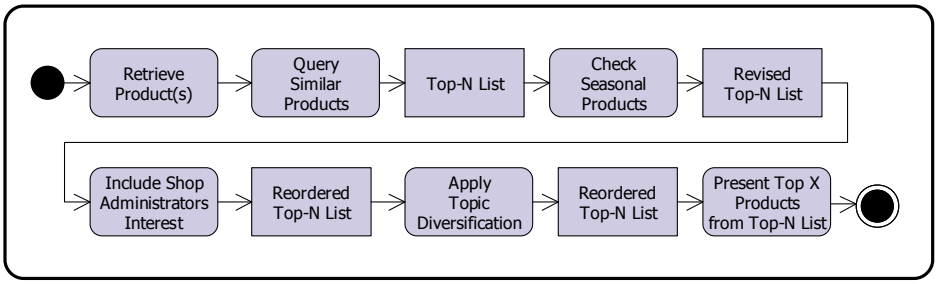

Figure 3 The process for generating a top-n list

The process starts by retrieving the products on which the recommendation is based. These are either the old products that have been clicked, reviewed or bought by a customer, or the products in the cart of the customer, or a combination of both. It has been shown that a first top-n list can be retrieved from the database containing the product similarities with only one SQL statement. After the top-n list is obtained, seasonal products could be filtered out. This reduces the top-n list to products that should be recommended within the current season. Afterwards, the shop administrator's interests could be used to reorder the top-n list. Finally, topic diversification could be used to reorder the top-n list a second time. The resulting top-n list can be shown to the customer.

Integration of a Recommender System into an Online Shop

Integrating a Recommender System in an existing online shop could be done for one concrete instance or generally. In the latter case, the developers of the online shop system need to define attributes that enable their customers to fine-tune a recommender system. In several workshops, the authors and industry partners identified three attributes that could be included in an online shop recommender system in order to fine-tune the recommendations:

1. Product's Half-Life: What is the maximum age of a product that should be recommended?
2. Season: What is the low and high season of a product (if it exists)?
3. Shop Interests: An attribute to push single products or to exclude them from the recommendations list.

The product's half-life is an interesting attribute to limit the recommendations only to newer products. This leads to the effect that products slide down or disappear from the recommendations list automatically after they reached a specified age.

A high seasonality of the products requires a parameter to specify the period (e.g. high or low season, month, Easter, Christmas) in which a product has to be recommended. This will avoid the recommendation of a winter glove in summer and other inappropriate suggestions.

In order to realize selling strategies, suppliers may wish to push single products using the recommender system (e.g. to promote a product or to dispose of slow sellers). Or they want single products to be excluded from the recommendations list. To enable the manipulation of recommendations either way, a parameter is needed which could be created as a slider. The slider shifts the product's position in the recommendations list to foster or lower the sales of this specific product.

The list of presented attributes to optimize recommendations is not comprehensive. More research has to be done and more experiences have to be collected in this area.

Conclusion and Outlook

The analysis of recent collaborative filtering techniques and different alternatives to optimize recommendations shows that the implementation and optimization of a collaborative recommender system for online shops is an extensive process. To decide which kind of optimizations could suitably be integrated in a concrete case, a reference process is introduced. The described reference process should be a valuable means to facilitate the implementation and optimization of a recommender system in an online shop. The selecting and adjusting of the relevant parameters should be supported and the process of parameterization should be automated.

The next step will be to validate the presented statements by practical implementations of recommender systems. The experiences from these implementations will lead to further insights regarding the coordination of optimization techniques. More research is needed to develop instruments for adapting a general recommender system to a specific case. Three relevant aspects have been presented. Now, further work is needed to find out how the adaptation could be simplified – not only for developers but also for the operating company.

Acknowledgements

The authors would like to thank all persons involved in the PersECA II project. Without them, this paper would not have been possible. Additionally, we owe special thanks to the Swiss KTI/CTI and the Swiss Federal Office for Professional Education and Technology who are funding this project under grant No. 8092.2 ESPP-ES.

References

Adomavicius, Gediminas and Alexander Tuzhilin, Towards the next gener¬ation of recommender systems: A survey of the state-of-the-art and possible extentions. IEEE Transactions on Knowledge and Data Engineering, 17(6):734–749, 2005.

Basu, Chumki, Haym Hirsh, and William Cohen, Recommendation as clas¬sification: Using social and content-based information in recommendation. In Proceedings of the 1998 workshop on recommender systems, pages 11–15, 1998.

Breese, John S., David Heckerman, and Carl Kadie, Empirical Analysis of Predictive Algorithms for Collaborative Filtering. Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, 1998.

Burke, Robin, Hybrid recommender systems, survey and experiments. User Modeling and User Adapted Interaction, 12(4):331–370, 2002.

Chao, Dennis L., Justin Balthrop, and Stephanie Forrest, Adaptive radio: Achieving consensus using negative preferences. In Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work, pages 120–123, 2005.

Deitel, Harvey M., Paul J. Deitel, and Kate Steinbuhler, E-Business and E-Commerce for Managers. Prentice Hall, 2001.

Deshpande, Mukund and George Karypis, Item-based top-n recommenda¬tion algorithms. ACM Transactions on Information Systems, 22(1):143–177, 2004.

Govers, Pascalle C.M. and Jan P.L. Schoormans, Product personality and its influence on consumer preference. Journal of Consumer Marketing, 22(4):189–197, 2005.

Hofmann, Thomas, Latent Semantic Models for Collaborative Filtering. ACM Transactions on Information Systems, 22 (1):89-115, 2004.

Herlocker, Jonathan L., Joseph A. Konstan, Loren G. Terveen, and John T. Riedl, Evaluating collaborative filtering recommender systems. ACM Trans¬actions on Information Systems (TOIS), 22(1):5–53, 2004.

Ling, Kimberly, Gerard Beenen, Pamela Ludford, Xiaoqing Wang, Klarissa Chang, Xin Li, Dan Frankowski, Loren Terveen, Al Mamunur Rashid, Paul Resnick, and Robert E. Kraut, Using social psychology to motivate contributions to online communities. Journal of Computer-Mediated Com¬munication, 10(4), 2005.

Lemire, Daniel and Anna Maclachlan, Slope one predictors for online rating-based collaborative filtering. In Proceedings of the SIAM Data Mining (SDM), 2005.

Linden, Greg, Brent Smith, and Jeremy York, Amazon.com recommenda¬tions. IEEE Internet Computing, 3:76–80, February 2003.

McCarthy, Joseph F. and Theodore D Anagnost, Musicfx: An arbiter of group preferences for computer supported collaborative workouts. In Pro¬ceedings of the ACM Conference on Computer Supported Cooperative Work, pages 363–372, 1998.

Manber, Udi, Ash Patel, and John Robison, The business of personalization: Experience with personalization of yahoo! Communications of the ACM, 43(8):35–39, 2000.

Resnick, Paul, Neophytos Iacovou, Mitesh Suchack, Peter Bergstrom, and John Riedl, Grouplens: An open architecture for collaborative filtering of netnews. In Proceedings of the ACM Conference on Computer Supported Cooperative Work, 1994.

Reichwald, Ralf, Frank T. Piller, and Kathrin Möslein, Mass customization concepts for the economy — four strategies to create competitive adantage with customized goods and services on the internet. In Proceedings of the Workshop on Information Systems for Mass Customization (ISMC), 2000.

Schubert, Petra and Michael Koch, The power of personalization: Customer collaboration and virtual communities. In Proceedings of the 8th Americas Conference on Information Systems (AMCIS), 2002.

Sarwar, Badrul, George Karypis, Joseph Konstan, and John Riedl, Analysis of recommendation algorithms for ecommerce. In Proceedings of the ACM Conference for Electronic Commerce, 2000.

Smyth, Barry and Paul McClave, Similarity vs. diversity. In: Proceedings of the 4th International Conference on Case-Based Reasoning, 2001.

Stormer, Henrik, Daniel Risch, and Nicolas Werro, Recommender systems for SME online shops: An Experiment. In Proceedings of the We-B Conference, 2006.

Weng, Sung-Shun and Mei-Ju Liu, Personalized product recommendation in e-commerce. In Proceedings of the IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE), pages 413–420, 2004.

Ziegler, Cai-Nicolas, Georg Lausen, and Lars Schmidt-Thieme, Taxonomy-driven computation of product recommendations. In Proceedings of the 2004 ACM CIKM Conference on Information and Knowledge Management, pages 406–415, 2004.

Ziegler, Cai-Nicolas, Sean M. McNee, Joseph A. Konstan, and Georg Lausen, Improving recommendation lists through topic diversification. In Proceedings of the 14th international conference on World Wide Web (WWW), 2005.

View publication stats