Improving E-Commerce Recommender Systems by the Identification of Seasonal Products

Henrik Stormer
Department of Informatics
University of Fribourg
Bd de Perolles 90
1700 Fribourg
Switzerland
henrik.stormer@unifr.ch

Abstract

In recent years, the number of recommender systems used in online shops has strongly increased and is becoming an important success factor for electronic commerce applications. A recommender system can be utilized to suggest similar related or potentially interesting products for a given customer or a set of products for a marketing campaign. This paper shows how seasonal products can be identified and included in the recommendation process to improve the quality of a recommender system. The approach was tested using real life data from two companies selling working protection and tools.

Introduction

In recent years, recommender systems have been vastly improved and are becoming an important success factor for a number of companies. A recommender system is a personalized information filtering technology that can be used to either identify a set of products that will be of interest to a certain user or to predict whether a particular user will like a particular product. Recommender systems have been applied to different areas, for example in the field of books (Linden, Smith, & York 2003), music (McCarthy & Anagnost 1998; Chao, Balthrop, & Forrest 2005) and movies (Ling *et al.* 2005).

This paper concentrates on recommender systems for electronic commerce. In this case, similar related or potentially interesting products are suggested for a given customer. A number of online shops already integrated a recommender system in their web appearance, well known examples are Amazon (Linden, Smith, & York 2003) or CDNow (Schafer, Konstan, & Riedl 1999).

The optimization presented in this paper is based on the identification of seasonal products. Seasonal products are products that are sold mostly during a part of the year, for example in winter. Today's E-commerce recommender systems do typically not include seasonal information. This means, that a seasonal product like a winter coat could be recommended to a customer in summer. It is obvious that such a recommendation will lead the customer to question the quality of the system in general. This paper shows that including seasonal information is a promising way to improve the quality of a recommender system.

The remainder of the paper is divided into 4 sections and is structured as follows: The next Section provides an introduction to recommender systems. Section 3 shows how seasonal products can be identified using transaction data. Section 4 evaluates the proposed method using two different real shop data sets. Finally, Section 5 concludes the paper and presents an outlook.

Recommender Systems

The purpose of recommender systems is to recommend products according to the users preferences. The broad area of recommender systems was introduced in the mid-1990s by some early papers on collaborative filtering (Resnick *et al.* 1994). Since then the term recommender system has become more common because it comprises content-based filtering, collaborative filtering as well as hybrid approaches.

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, Hirsh, & Cohen 1998). The advantage of content-based filtering is the possibility of precisely defining relations between products, namely for the purpose of 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. This information should always be up-to-date. On the other hand, the content-based filtering uses complicated data mining techniques to generate the personalized information.

In contrast to content-based filtering, the collaborative filtering approach only needs information about the users' interaction and transaction such as products ratings, orders or clickstream information in order to provide recommendations. All of this information is continuously provided by the users when browsing the websites, buying or rating products. 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, Heckerman, & Kadie 1998) or statistical techniques (Hofmann 2004).

The algorithm presented in this paper falls in the collaborative filtering and model-based classes. The collaborative filtering approach can be implemented by 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. 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. The following Table shows an example containing three products and three users:

	DVD Lost in Translation	**DVD The Bourne Supremacy**	**DVD The Life Aquatic**
Mr. Smith	$\emptyset$	$+0.5$	$\emptyset$
Mr. Johnson	-0.5	-1	$+0,5$
Mrs. Miller	$+1$	$+0.5$	$\emptyset$

In the example, *Mr. Miller* is a big fan of the product *DVD Lost in Translation* and rated it with the highest value ($+1$). However, Mr. Johnson does not like the product and therefore rated it low (-0.5).

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 new matrix with the customers on both dimensions and the similarities as entries is returned:

	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 to assign each user to a 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 hasn't 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 the products. Once again, several mathematical approaches can be used to calculate these similarities. In this paper, the methodology of (Deshpande & Karypis 2004) has been chosen.

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

Seasonal Products

What are Seasonal Products?

The problem of seasonal products, that are highly desirable in one month but of no interest in another month has been presented by other researchers (Schafer, Konstan, & Riedl 2001). However, up to our knowledge, no approach exist that shows how to include seasonal products in a recommender system. For this task, two main points have to be fulfilled:

1. A seasonal product has to be identified.
2. A recommender system should include only the products that are sold during the current season.

The identification of a seasonal product (the first step) can be done automatically or manually. In the latter case, an administrator has to define seasonal periods, in which the product should be promoted. Typically, a seasonal product has a high and a low season. In the high season, most of the sales are made whereas in the low season, customers do not order the product.

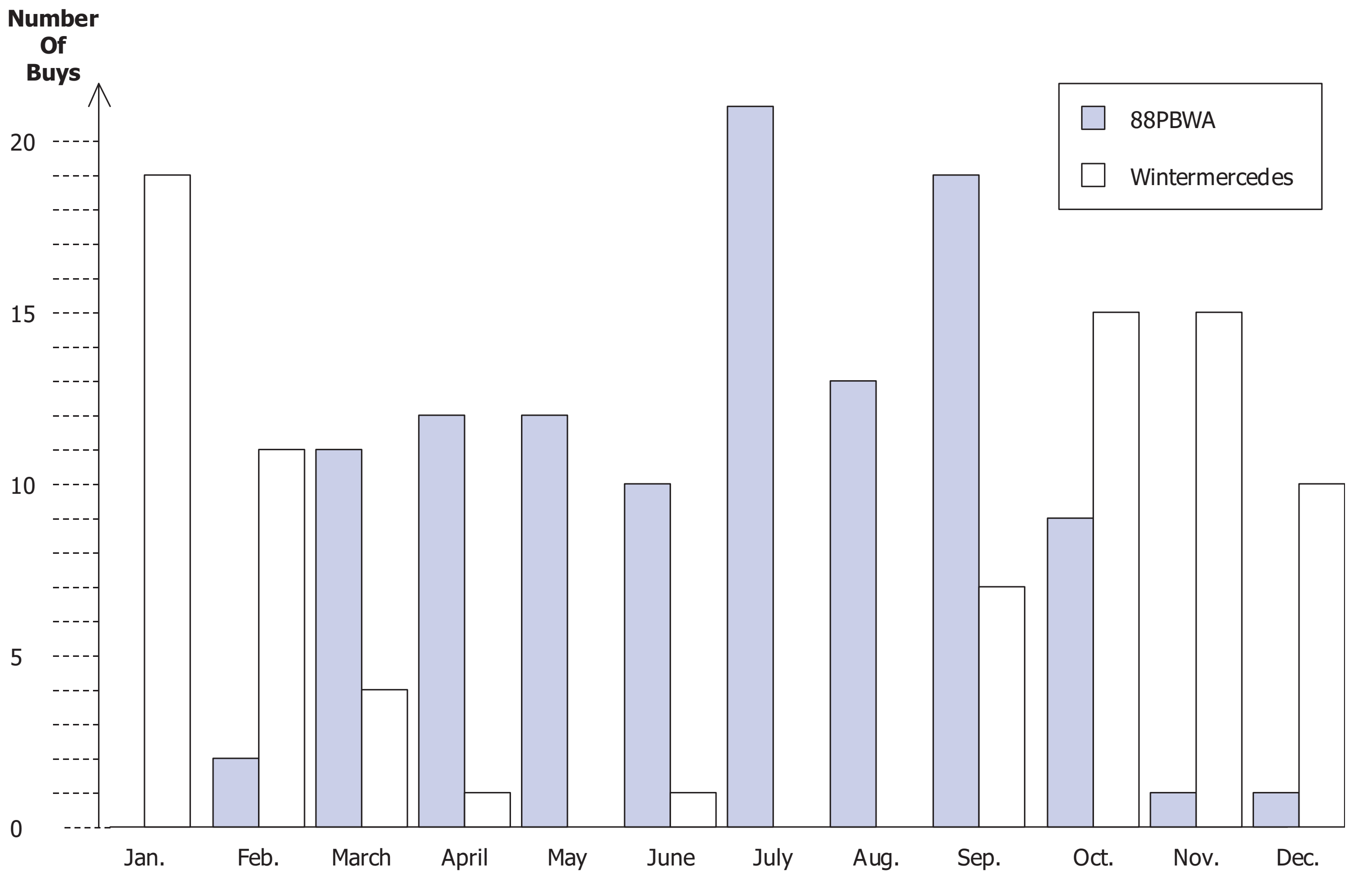

Figure 1: Two seasonal products.

Our tests have shown that the automatic identification of the high and low season (if there are seasons) is harder than the second part, because once the low season has been retrieved for all seasonal products, a product list can be changed by removing all products with a low season. The next section deals with the problem of identifying a seasonal product.

Identifying Seasonal Products

Figure 1 shows two seasonal products. The first product, called *Wintermercedes*, is a working glove that is, as the name indicates, mostly used in winter times. Figure 1 shows that the high season of the glove is within the period of September to March. The second product is another working glove, called *88PBWA*. This product is typically sold during summer, namely within the period of March to October.

To calculate the seasonal influences, a number of parameters could be evaluated:

Quantities: The number of sold quantities during one month, as shown in Figure 1.

Turnover: The turnover of the product within one month.

Different Users: The number of different users that bought the product during one month.

If the price of the product has not changed during the last year, quantities and turnover are similar. The parameters can easily be retrieved from the online shop or ERP database. Typically, the last twelve months are used, if older data is available, it should also be included. However, care must be taken to not tamper the data by including some months more often than others.

For our tests, we used the ordered quantities as shown in Figure 1. The function $s_i(m_j)$ returns the sold quantities of product i for month m_j. For example, product Wintermercedes presented in Figure 1 has sold 15 times in November, therefore $s_{Wintermercedes}(11) = 15$.

Although it has been mentioned that identifying seasonal products could improve a recommender system (Schafer, Konstan, & Riedl 2001), there exist little work in this area. In economics, seasonal influences are seen as problematic and it has been attempted to remove it from the data (Zellner 1979) or to find strategies that include seasonality (Kawasaki 2006). For the determination of seasonal products, a number of marketing approaches have been described (McMath 1994). One example is the work of Radas and Shugan (Radas & Shugan 1998), who provide a complex algorithm using mathematical estimation functions to determine seasonal products. However, their approach aims to predict the future sales of the product whereas for a recommender system, it is sufficient to find the low and high seasons of a product (if they exist).

In our test, a simple approach was used that calculates the ratio between orders in one month compared to all orders. For each month, we summed up the quantities from this month, the preceding and the following month. Then, the ratio to all orders was calculated. Figure 1 shows that product Wintermercedes has been sold 83 times in 2006. To calculate the ratio for July, the quantities of June (1 product sold), July (0 products sold) and August (0 products sold) were added to a total of 1 product sold within the three month. The ratio is $1/83$ or 1.2% of all sales. The month ratio (MR) is therefore calculated using formula:

$$mr(p_i, m_j) = \frac{s_i(m_j - 1) + s_i(m_j) + s_i(m_j + 1)}{\sum_{k=1}^{12} s_i(m_k)}$$

where p_i denotes product i and the denominator calculates the sum of all orders for product i within the last twelve months, for example $\sum_{k=1}^{12} s_{Wintermercedes}(m_k) = 83$.

This calculation can be done for each product ($1 \leq i \leq |P|$) and each month ($1 \leq j \leq 12$). However, it turned out that two parameters should be defined to optimize the calculation:

Minimum Order Threshold (MOR): The minimum order threshold defines the number of minimum orders that are needed to analyze the product. If a product has been sold only very few times within the last 12 month, a prediction of whether this product is a seasonal one and identifying the low and high seasons is not possible.

Ratio Threshold (RT): The ratio threshold defines for which months a product should be removed from the result set. In the example above, product Wintermercedes would be removed in July if the ratio threshold is lower or equal to 0.012.

An Algorithm for Seasonal Product Filtering

Using both parameters, the following algorithm is proposed to improve a recommender system by regarding seasonal dependencies:

```
INCLUDESEASONALDEPENDENCIES($topN, month$)
1  forall $p_i \leftarrow topN$
2    do
3      if $s_i >= MOR$
4        then
5          if MR($p_i, month$) $< RT$
6            then
7              REMOVEFROMLIST($topN, p_i$)
```

The function expects two input parameters: The first one called *topN* ($topN = p_1, p_2, \ldots, p_n$) denotes the top-N list retrieved by a standard recommender system. The second one called *month* ($1 \leq month \leq 12$) gives the month at which the top-N list should be calculated. The algorithm checks if the minimum order threshold is equal or greater then the number of orders for the product. If this is the case, the function MR is called to calculate the month ratio. If the ratio is lower than the ratio threshold, the product is removed from the top-N list. The function *RemoveFromList* deletes product p_i from the top-N list, this is not described in this paper.

Evaluation

For the evaluation, two different data sets were used. One from the German company Kiel & Co. GmbH `www.kielundco.de`, the other from the Swiss company Bruetsch Ruegger AG `www.brw.ch`. Kiel & Co. is offering working protection like gloves, trousers and shoes; Bruetsch Ruegger is selling tools like drills, micrometers and screwdrivers. The data sets have the following number of elements:

	Number of Users	Number of Products	Number of Transactions
Bruetsch Ruegger	$13'420$	$8'064$	$385'610$
Kiel & Co.	$2'367$	$1'153$	$6'707$

The Kiel & Co. data set is much smaller compared to Bruetsch Ruegger. Especially the relatively small number of transactions is responsible for problems when identifying seasonal products.

We started by analyzing the impact of the minimum order and ratio thresholds and ran a number of tests on both data sets. The results are presented in Table 1. Using a constant ratio threshold of 0.001, the minimum order threshold was changed from 4 orders (all products with less than 4 buys within the last year would not be included) to 20. Obviously, the number of removed products is decreased and the number of products below the threshold is increased. With a border of 20, almost half of the products from Bruetsch Ruegger would not be included in the calculation because they have less than 20 buys. For Kiel & Co., more than two third of all products would not be included in the calculation if the minimum order quantity is 4. Additionally, the number of months a product would be removed shrinks from 5508 to 1658 for Bruetsch Ruegger and from 792 to 33 for Kiel & Co.

Another test has been done with different ratio and a constant minimum order threshold of 4. With a ratio threshold of 0.1(=10%), almost half of the products would be removed from the result set at least in one month for Bruetsch Ruegger whereas for Kiel & Co. only 300 products would be removed at least once. It is important to note that the reason for this low value lies in the fact that the data set of Kiel & Co. would not consider 736 of all products.

Afterwards, the algorithm was evaluated by splitting the data sets of both transaction data in two parts: a test set containing 500 randomly selected non-zero entries of the rating matrix and a training set with the remaining entries.

This approach is well known and used in by a number of other evaluations (Herlocker *et al.* 2004). The algorithm was run using the training set. Afterwards, the quality of the algorithm was measured by looking at the test set elements. For each test set element of user i that bought product j, we calculated the top-N recommendations for i. In our test set,

Minimum Order Threshold	Products below Threshold	Products Removed at least Once	Number of Removed Months
4	1′312	1′665	5′508
6	1′615	1′369	4′248
8	1′863	1′150	3′498
10	2′085	972	2′873
12	2′274	866	2′552
14	2′447	764	2′274
16	2′611	667	1′985
18	2′754	611	1′861
20	2′908	547	1′658
Ratio Threshold	**Products below Threshold**	**Products Removed at least Once**	**Number of Removed Months**
0.001	1′312	1′665	5′508
0.005	1′312	1′669	5′520
0.01	1′312	1′685	5′545
0.05	1′312	2′173	6′688
0.1	1′312	3′483	10′433

Minimum Order Threshold	Products below Threshold	Products Removed at least Once	Number of Removed Months
4	736	220	792
6	818	141	429
8	872	91	250
10	905	66	153
12	930	50	110
14	956	34	56
16	978	24	44
18	989	21	39
20	998	18	33
Ratio Threshold	**Products below Threshold**	**Products Removed at least Once**	**Number of Removed Months**
0.001	736	220	792
0.005	736	220	792
0.01	736	221	793
0.05	736	247	862
0.1	736	300	1′070

Table 1: Changing the minimum order and ratio threshold has an impact on the result set of Bruetsch Ruegger (top) as well as Kiel & Co. (bottom).

N was always set to 20. If product j was within the top 20 products, we called this element a hit. If not, we called it a miss. The average hit-rate (AHR) can be calculated using formula:

$$ahr = \frac{\text{Number of hits}}{\text{Number of elements in test set}}$$

As already stated, the test set contained 500 elements and therefore the denominator was set to 500.

To better weight the position of a hit within the recommendation list, a number of researchers proposed to calculate the average reciprocal hit rank (ARHR) (i.e. (Deshpande & Karypis 2004)), that is calculated using formula:

$$arhr = \frac{1}{n}\sum_{i=1}^{h}\frac{1}{p_i}$$

where p_i is the position of hit i. If the hit is always on the first position of the top-N list, ARHR is equal to AHR. Otherwise, ARHR is smaller because a hit at position i is not weighted by 1 but by $1/i$.

Within the test, we calculated AHR and ARHR without including seasonal dependencies first. For the calculation of product similarities, the well known approach by Deshpande and Karypis was used (Deshpande & Karypis 2004). Deshpande and Karypis define the number of different customers that bought a product as $Freq$, thus $Freq(i)$ is the number of nonempty entries in the rating matrix for product's i vector. The following formula is used to calculate the equality between two product options p_i and p_j :

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

where $M_{i,j}$ is cell (i, j) of the rating matrix and α a parameter between 0 and 1. According to the tests from (Deshpande & Karypis 2004), a good value for α is between 0.3 and 0.6. The calculation was done with α set to 0.5.

In order to ensure that the results are not sensitive to the particular test set, the calculation was performed with ten different runs, each time using a different random partitioning into training and test set:

	AHR	**ARHR**
Bruetsch Ruegger	$0.151 (= 15.1\%)$	0.0509
Kiel & Co.	$0.62 (= 62\%)$	0.2705

Afterwards, the test was changed. For each element of the test set, we extracted the month it was bought. Then, we alternated the ratio threshold with values of 0.1, 0.01, and 0.001 and the minimum order threshold with values of 4, 10, and 16 as described in section (MOT and RT denote minimum order threshold resp. ratio threshold):

	Bruetsch Ruegger	**Kiel & Co.**
MOT=4 & **RT**=0.1	AHR=0.171 ARHR=0.071	AHR=0.629 ARHR=0.276
MOT=4 & **RT**=0.01	AHR=0.179 ARHR=0.072	AHR=0.63 ARHR=0.272
MOT=4 & **RT**=0.001	AHR=0.177 ARHR=0.072	AHR=0.63 ARHR=0.272
MOT=10 & **RT**=0.1	AHR=0.192 ARHR=0.072	AHR=0.622 ARHR=0.271
MOT=10 & **RT**=0.01	AHR=0.189 ARHR=0.069	AHR=0.622 ARHR=0.271
MOT=10 & **RT**=0.001	AHR=0.187 ARHR=0.065	AHR=0.621 ARHR=0.2709
MOT=16 & **RT**=0.1	AHR=0.1514 ARHR=0.052	AHR=0.62 ARHR=0.2708
MOT=16 & **RT**=0.01	AHR=0.151 ARHR=0.051	AHR=0.62 ARHR=0.2706
MOT=16 & **RT**=0.001	AHR=0.151 ARHR=0.051	AHR=0.62 ARHR=0.2705

In the best case (MOT=10 and RT=0.1), the hit rate of Bruetsch Ruegger improved from 15.1% to 19.2%. This is an improvement of 4.1%. The reciprocal hit rate improved by 0.02.

For Kiel & Co., as the data set is much smaller, the best case is with MOT=4 and RT=0.01 (improvement of around 1%). ARHR also improved only marginally (around 0.006).

It can be seen from the results that with a high value of MOR and a low value of RT, the influence on the result is marginally. This is because less seasonal products are identified. However, if the MOR value is too low, possible hits are also removed. This can discovered by comparing the values MOT=4 and RT=0.1 with MOT=10 and RT=0.1 for Bruetsch Ruegger.

It is also important to note that all values are better than the calculation without seasonal filtering. Of course, be choosing a higher value of RT than 0.1 or a lower value of MOT than 4, it is possible to create a scenario in which the seasonal filtering returns worse results.

Computation

An important aspect of the presented method is the inclusion in existing recommender systems. It is possible to divide this task in two parts. In the offline part, the inclusions of each product in month j are calculated using a similar function as *IncludeSeasonalSimilarities* defined above:

```
OfflineCalculation()
1 forall $p_i \leftarrow P$
2   do
3     if $s_i >= MOR$
4       then
5         for $j \leftarrow 1$ to 12
6         if MR$(p_i, j) < RT$
7           then
8             MarkProduct$(p_i, j)$
```

Instead of checking only the product from the topN list, all products are included by iterating over the set of products P. Then, for each month ($1 \leq j \leq 12$), it is evaluated if product p_i would be removed in month j. If this is the case, function *MarkProduct* is called. This function could store the result in a database, for example by creating a table called *seasonalCalcuation* containing columns for the product (*productId*) and boolean values for all 12 months (*month1,...,month12*).

In the online part, the results from the offline calculation can be used to remove the products from the topN list effcently. If the topN list for the product with ID 15 should be retrieved in December, the following example SQL Statement could be used:

```
select productId2 from itemSimilarity,
                       seasonalCalculation

  where itemSimilarity.productId2=
        seasonalCalculation.productId

  and itemSimilarity.productId1=15

  and seasonalCalculation.month12=true

  order by itemSimilarity.similarity
  desc
```

This command assumes the database table *itemSimilarity* containing for each product (*productId1*) the similiarities (*similarity*) to all other products (*productId2*). The results from the seasonalCalculation are joined with this table to delete all products that would be excluded.

This approach has been implemented and tested. For both data sets, the performance did not decrease as most of the calulation is done offline.

Discussion and Outlook

This paper showed a way to determine seasonal products and showed that removing a seasonal product during a low season improves a real life recommender algorithm. The main advantage of the presented approach is its simplicity which makes it easy to integrate in existing E-commerce recommender systems.

The main drawback of the presented approach is the historical transaction data needed to determine a seasonal product. If the product is new or was not sold well during the last year the determination is not possible.

In interesting idea to improve the determination of seasonal products is the usage of a content-based approach to find products with special properties, for example by extracting keyword from the product description. Term like 'winter', 'christmas' or 'sun' could indicate possible seasonal products. Additionally, if two products are similar according to their properties and one is a seasonal product where the low season could be determined, it is likely that the other one has a similar low season. However, this assumption needs to be proofed.

We are planning to integrate the described seasonal filtering in the online shop of Bruetsch Ruegger. They already have a recommender system that can easily be extended. We also think that filtering seasonal products improves the opinion of a customer towards the recommender system as seasonal products are not recommended in the low season.

Acknowledgements

The author wishes to thank all persons involved in the PersECA II project. Without them, this paper would not have been possible. Additionally, thanks to the companies Bruetsch Ruegger AG and Kiel & Co. GmbH for providing the datasets needed for the evaluatation.

References

Adomavicius, G., and Tuzhilin, A. 2005. Towards the next generation 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.

Basu, C.; Hirsh, H.; and Cohen, W. 1998. Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the 1998 workshop on recommender systems*, 11–15.

Breese, J. S.; Heckerman, D.; and Kadie, C. 1998. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*.

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

Chao, D. L.; Balthrop, J.; and Forrest, S. 2005. Adaptive radio: Achieving consensus using negative preferences. In *Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work*, 120–123.

Deshpande, M., and Karypis, G. 2004. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems* 22(1):143–177.

Herlocker, J. L.; Konstan, J. A.; Terveen, L. G.; and Riedl, J. T. 2004. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)* 22(1):5–53.

Hofmann, T. 2004. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems* 22(1):89–115.

Kawasaki, Y. 2006. A structural time series model facilitating flexible seasonality. In *Proceedings of the Conference on Seasonality, Seasonal Adjustment and Their Implications for Short-term Analysis and Forecasting*.

Linden, G.; Smith, B.; and York, J. 2003. Amazon.com recommendations. *IEEE Internet Computing* 3:76–80.

Ling, K.; Beenen, G.; Ludford, P.; Wang, X.; Chang, K.; ; Li, X.; Frankowski, D.; Terveen, L.; Rashid, A. M.; Resnick, P.; and Kraut, R. E. 2005. Using social psychology to motivate contributions to online communities. *Journal of Computer-Mediated Communication* 10(4).

McCarthy, J. F., and Anagnost, T. D. 1998. Musicfx: An arbiter of group preferences for computer supported collaborative workouts. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, 363–372.

McMath, R. 1994. Product proliferation. *Mediaweek* 4:34–35.

Radas, S., and Shugan, S. M. 1998. Seasonal marketing and timing new product introductions. *Journal of Marketing Research* 35(3):296–315.

Resnick, P.; Iacovou, N.; Suchack, M.; Bergstrom, P.; and Riedl, J. 1994. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*.

Sarwar, B.; Karypis, G.; Konstan, J.; and Riedl, J. 2000. Analysis of recommendation algorithms for ecommerce. In *Proceedings of the ACM Conference for Electronic Commerce*.

Schafer, J. B.; Konstan, J.; and Riedl, J. 1999. Recommender systems in e-commerce. In *Proceedings of the ACM Conference on Electronic Commerce*.

Schafer, J. B.; Konstan, J. A.; and Riedl, J. 2001. E-commerce recommendation applications. *Data Mining and Knowledge Discovery* 5(1–2):115–153.

Zellner, A., ed. 1979. *NBER-Census Conference on Seasonal Economic Time Series*. Government Printing Office.