ProMax: A Profit Maximizing Recommendation System for Market Baskets

by

Lydia Manikonda, Annu Tuli, Vikram Pudi

in

SIGAI Workshop on Emerging Research Trends in AI
(*ERTAI*)

Mumbai, India

Report No: IIIT/TR/2010/39

Centre for Data Engineering
International Institute of Information Technology
Hyderabad - 500 032, INDIA
April 2010

ProMax: A Profit Maximizing Recommendation System for Market Baskets

Lydia Manikonda Annu Tuli Vikram Pudi

***Abstract*— Most data mining research has focused on developing algorithms to discover statistically significant patterns in large datasets. However, utilizing the resulting patterns in decision support is more of an art, and the utility of such patterns is often questioned. In this paper we formalize a technique that utilizes data mining concepts to recommend an optimal set of items to customers in a retail store based on the contents of their market baskets. The recommended set of items maximizes the expected profit of the store and is decided based on patterns learnt from past transactions. In addition to concepts of clustering and frequent itemsets, the proposed method also combines the idea of knapsack problems to decide on the items to recommend. We empirically compare our approach with existing methods on both real and synthetic datasets and show that our method yields better profits while being faster and simpler.**

I. Introduction

The data mining research literature is replete with several algorithms – most of which are elegant, efficient and effective. However, there are only a few formal studies concerning how data mining can actually be beneficial in more specific targets. A major obstacle in data mining application is the gap between statistically significant pattern extraction and value-based decision making [2]. It is often questioned as to how exactly one should make use of the patterns extracted through data mining algorithms for making effective business decisions, with the ultimate goal of yielding better profits for the business.

Similarly, studies about the retail market [1] have received wide attention, although only a few of them have seriously dealt with data mining. Ke Wang et al. [2] first presented a profit mining approach to reduce this gap in 2002 and recent investigations have shown an increasing interest on how to make decisions by utilizing association rules. In this paper we focus on the problem of recommending products to customers in retail stores such that the profit of the store is maximized.

Market basket databases contain historical data on prior customer choices where each customer has selected a subset of items, a market basket, out of a large but finite set. This data can be used to generate a dynamic recommendation of new items to a customer who is in the process of making the item choices. Some retail outfits provide carts with displays that provide product information and recommendations as the customer shops. Remote shopping systems allow customers to compose shopping lists through personal digital assistants (PDAs), with interactive recommendations of likely items. Internet commercial sites often provide dynamic product choices as the customer adds items into the virtual shopping cart, or market basket. Internet sites also display dynamically changing set of links to related sites depending on the browsing pattern during a surfing session. Faced with an enormous variety of options, customers surfing the web gravitate toward sites that offer information tailored to their personal preferences. All these activities are characterized by the progressive item choices being made by a customer, and the providers' desire to recommend items that are the most likely next choice of the customer [14].

From the market angle, two important criteria [5] should be taken into account during the process of mining profit: the items in retail shops should first meet the basic sale request, and second, should bring higher profits. Therefore, how to meet these two principles is the core problem of profit mining. The cross-selling effect of items [1] has been noticed by current retailers: the profit of an item is not only involved in the item itself, but is also influenced by its related items. Some items fail to produce high profit by themselves, but they might stimulate customers to buy other profitable items. Consequently, the cross-selling factor which can be studied by the analysis of historical transactions should be involved in the problem of item selection.

Searching for such a relation of items to support cross-selling has become an important issue. Current approaches to study these relationships are based on association rules. However, association rules by themselves do not suggest how to maximize profit.

In this paper, we present an algorithm that combines simple ideas from clustering, recommendation systems, and the knapsack problem to recommend those items to customers that maximize the expected profit. We tested our algorithm on two popular datasets: One was generated by using the data generator from IBM Almaden Quest research group [13] and the other was a retail market basket dataset available on the FIMI[1] repository. Our experiments show that our algorithm is performing better than the competing algorithms in terms of obtaining better profits, while at the same time being faster and simpler.

This paper is organized as follows: In Section 2, we introduce the related work of profit mining. In Section 3, we describe the problem statement. In section 4 we discuss our algorithm. Detailed experimental results are presented in Section 5. In Section 6, we draw conclusions and present

Lydia Manikonda, Annu Tuli and Vikram Pudi are with the Center for Data Engineering, International Institute of Information Technology, Hyderabad, India (email: {{lydia, annu}@research., vikram}@iiit.ac.in).

[1]http://fimi.cs.helsinki.fi/data

future work.

II. Related Work

Many novel and important methods were proposed to support profit mining. In 2002, profit mining approach and other related problems [2], [6] were presented by Ke Wang et al. The webpage-layered algorithm HITS [8] was extended as HAP algorithm [7] by Ke Wang et al, to solve the problem of item ranking with the consideration of the influence of confidence and profit, which has still several drawbacks [9]. In order to mine a subset of items with the maximal profit while improving the above drawbacks, the maximal profit problem of item selection (MPIS) [9] was proposed by Raymond Wong et al. However, MPIS is too difficult to implement and solves a NP-hard problem even in the simplest situation. In other words, although MPIS algorithm could find the best solution, the cost of time is too expressive to be tolerated.

By considering the cross-selling effect in order to solve the problem of Item Selection for Marketing (ISM), Hill Climbing method [4] was recently proposed by Raymond Wong et al. Raymond Wong et al. [10] also adopted genetic algorithms to generate local optimal profit to fit the optimal profit of the item selection problem.

The DualRank [12] algorithm which uses a graph based on association rules and is compressed because the number of out-degrees for the items decrease if the minimum support is increased. In general if the support of items is decreased then the number of association rules will keep increasing. Matrix calculations being done are affected which then affects the item selection based on the profit. Another problem with DualRank [12] is that it is not very efficient for sparse data sets and it includes cumbersome calculations of eigenvalues and eigenvectors which become intractable for large transactional data sets. To overcome all these situations we have developed in this paper a new algorithm for market basket datasets.

III. Problem Definition

In this paper we focus on the problem of recommending products to customers in retail stores such that the profit of the store is maximized. The following definition formally captures the problem statement:

Definition 1 (Profit Mining): Given a transactional dataset $D = \{t_1, t_2, \ldots, t_m\}$, where each transaction t_i contains a subset of items (itemset) from the set $I = \{i_1, i_2, \ldots, i_n\}$, each item i_j having an associated profit p_j, the problem is to select k additional items to recommend for each transaction, with the goal of making more sales and maximizing the total profit of the resulting transactions.

There are several challenges to this problem:

1) **Model product purchase probabilities:** We need to recommended products that are more likely to be purchased. It is therefore important to have a clear understanding of how to model the probability of purchase of each product.
2) **Model product relationships:** It is not sufficient to know the individual purchase probabilities of different products. We need to also identify relationships between items for cross-selling. The standard technique in recent approaches for this purpose has been to use association rules.
3) **Model customers:** Even high-confidence association rules may not apply to a particular customer, whereas some low-confidence rules may apply. Thus, it is imperative to model customers regarding which category they belong to and then study the rules within that category. This is more likely to result in effective recommendations. The standard technique in the recommendation system community for this purpose is to cluster customers such that customers within each cluster share the same purchase patterns.
4) **Balance purchase probability and profit:** A pure association rule based approach will favor rules with high confidence so as to maximize the probability that the customer will purchase the recommended item. For example, the rule $\{Perfume\} \rightarrow \{Lipstick\}$ will be favored because of higher confidence compared to a rule $\{Perfume\} \rightarrow \{Diamond\}$. In contrast, a pure profit-based approach will favor the latter rule hoping for higher profit. Neither necessarily maximizes the true profit. Indeed, items of high profit often also have low supports because fewer people buy expensive stuffs.
5) **Decide the number of products to recommend:** An implicit constraint here is that if we recommend too many items then the customer will be overwhelmed by the choices and is likely to avoid choosing *any* item at all. On the other hand, if we recommend too few items, then we may miss a successful sale of a product that has not been brought to the attention of the customer. The correct number of products to recommend depends on the attention span of customers and would vary depending on the domain.

Clearly, the problem of recommending the right products for each market basket in a store is a challenging problem and thereby the task of designing an elegant, simple and yet effective algorithm seems very difficult upfront. In this paper we don't deal with issue 3 (model customers) directly. Instead we model customers using their transactions and hence we cluster transactions instead as mentioned in section 4.1. Also, we don't discuss the issue 5 because it is application dependent. In the experiments mentioned in section 5 we recommend five items to each customer.

IV. The ProMax Algorithm

As discussed in the previous section, at the face of it, the problem of recommending the right products is replete with several challenges. It therefore seems daunting to design a simple yet effective algorithm for this task, so much so that to design an *optimal* algorithm that guarantees to maximize the expected profit, seems out of reach. Yet this is exactly what we achieve. In this section we present **ProMax**, a *Pro*fit *Max*imizing recommendation system for market baskets.

Our algorithm performs a clustering of the customer transactions as a preprocessing. For this purpose, it uses the clustering algorithm in [11]. Next, at the time of recommendation, the algorithm is based on the following steps:

1) Identify the cluster C to which the current record belongs.
2) Calculate the expected profit of each item in the cluster C.
3) Sort the items based on expected profit and recommend the top k items, where k is a parameter to the algorithm.

First, in Section IV-A we describe the clustering algorithm used in step 1 and identifying the current cluster to which the user's transaction belongs to. Next, the manner in which expected profit of items is computed is later described in Section IV-B.

A. Clustering Customer Transactions and identification of the current cluster

Since the probability of earning more profit is directly proportional to the purchase probability of items, it is imperative to be able to accurately estimate the purchase probability of specific items by specific customers. A naive solution is to use the global support of items as an estimate of their probability. But, as customers differ in their purchase patterns, the global support of an item is an unreliable estimator of its likelihood of sale.

Therefore, a natural approach is to first cluster the transactions based on their purchase patterns and then use the support of items within a cluster as more accurate estimates of their purchase probabilities. The clustering criterion we use is based on the notion of *large items* and was proposed in [11].

For a given collection of transactions and a minimum support, our aim is to find a clustering C such that $cost(C)$ is minimum. $cost(C)$ is calculated by using the intra cluster similarity and inter cluster similarity which are calculated by using the notion of Small Items and Large Items respectively.

For a user-specified *minimum support* θ $(0 < \theta \leq 1)$, an item is *large* in cluster C_i if its support in C_i is atleast $\theta \times |C_i|$; otherwise, an item is small in C_i. Intuitively, large items are popular items in a cluster, thus, contribute to similarity of items within a cluster. In contrast, small items contribute to dissimilarity in a cluster. Let $Large_i$ denote the set of large items in C_i, and $Small_i$ denote the set of small items in C_i.

Consider a clustering $C = \{C_1, C_2, \ldots, C_k\}$. The cluster to which the current record r belongs to, depends on the cost which can be calculated using the equation 4. It can be defined mathematically as.

$$cluster(r) = argmin_i[Cost(C_i)] \tag{1}$$

The cost of C to be minimized depends on two components: the intra-cluster cost and the inter-cluster cost:

Intra-cluster cost: This component is charged for the intra-cluster dissimilarity, measured by the total number of small items:

$$Intra(C) = |\cup_{i=1}^{k} Small_i| \tag{2}$$

This component will restrain creating loosely bound clusters that have too many small items.

Inter-cluster cost: This component is charged for the inter-cluster similarity. Since large items contribute to similarity in a cluster, each cluster should have as little overlapping of large items as possible. This overlapping is measured as:

$$Inter(C) = \sum_{i=1}^{k} |Large_i| - |\ \cup_{i=1}^{k} Large_i| \tag{3}$$

In words, $Inter(C)$ measures the duplication of large items in different clusters. This component will restrain creating similar clusters.

To put the two together, one can specify weights for their relative importance. The cost function of the clustering C then is defined as:

$$Cost(C) = w \times Intra(C) + Inter(C) \tag{4}$$

A weight $w > 1$ gives more emphasis to the intra-cluster similarity, and a weight $w < 1$ gives more emphasis to the inter-cluster dissimilarity. By default $w = 1$.

The pseudo-code of the clustering algorithm described above is shown in Algorithm 1.

Algorithm 1 Clustering Algorithm

```
/*Allocation Phase*/
while not end of file do
  read the next transaction < t, − >;
  allocate t to an existing or a new cluster C_i to minimize
  Cost(C);
  write < t, C_i >;
end while
/*Refinement Phase*/
repeat
  not_moved = true;
  while not end of the file do
    read the next transaction < t, C_i >;
    move t to an existing non-singleton cluster C_j to
    minimize Cost(C);
    if C_i ≠ C_j then
      write < t, C_j >;
      not_moved = false;
      eliminate any empty cluster;
    end if
  end while
until not_moved ;
```

B. Calculating Expected Profit

Once the clusters are computed, the probability of an item's purchase can be found by first identifying which cluster the current transaction falls into and then looking up

the support of the corresponding item in that cluster. That is, the item's probability is estimated as its support within a cluster, rather than its global support. As mentioned in Section IV-A, this manner of computing the probability of items is far more accurate in terms of their likelihood of purchase.

In this context, we compute the *expected profit* of a given item i with the help of its probability (intra-cluster frequency) f and profit p. The *total expected profit* for n items can be computed as:

$$E_{total} = \sum_{i=1}^{n} f_i \times p_i \tag{5}$$

C. Recommending Items: Knapsack Approach

The current scenario is that we can recommend k items to the customer where k is a constant chosen based on the domain. The goal is to maximize profit – so we must ensure that the recommended items have high purchase probability *and* high profit.

There are mainly two options:

1) **Recommending items with high purchase probability:** Consider the example of lipstick and diamond. Suppose the cost of lipstick is \$1 and cost of diamond is \$500 and purchase probability of lipstick is 0.03 and diamond is 0.0001. When we consider the purchase probability as the only case, we will be recommending lipstick which would not yield high profit.
2) **Recommending items with high profit:** Consider the same example as given above, suppose the cost of lipstick is \$2 and cost of diamond is \$500, diamond has higher profitability than lipstick. When we consider only the high profit, then we will recommend diamond which is not a good recommendation.

If the goal was to maximize either purchase probability or profit separately as mentioned above, we could directly use the knapsack approach [3]. We reduce the current problem to the knapsack problem in the following manner.

Definition 2 (Knapsack Problem): Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than a given limit and the total value is as large as possible.

We reduce profit mining to a knapsack problem using the following equivalences:

1) Items to carry in a knapsack correspond to the items for sale in profit mining.
2) Item weight (in knapsack problem) are set to 1 for all items in the store.
3) Weight limit (in knapsack problem) is set to k – the number of items to recommend (in profit mining).
4) Item value (in knapsack problem) is set to expected profit of that item (which is purchase probability $\times$ item profit).

The result is a greedy algorithm to recommend items – Once the expected profit of items is computed, we sort all items in decreasing order of these expected profit values and recommend the k items which have the highest expected profits. The knapsack approach guarantees that this greedy approach maximizes the overall expected profit.

The pseudo-code of the resulting algorithm as described above is shown in Algorithm 2.

Algorithm 2 Mining Profitable Items

```
Identify the cluster to which the user's transaction belongs
I = items bought by the user
for not end of all the clusters do
    i = cluster id
    w = weight function
    C_i = i^th cluster
    Cost(C_i) = w × Intra(C) + Inter(C)
end for
min = minimum cost value
min_id = id of the cluster which has the minimum cost
while not end of the transactions in the cluster having min
cost do
    f = frequency of each item i
    p = profit of item i
    E_i = f_i × p_i (expected profit)
end while
sort the items in descending order of expected profits
select k items with maximum value from the sorted list
```

V. Experimental Evaluation

To evaluate the performance of the ProMax algorithm we ran a set of experiments on two data sets: a retail data set [5] available from the Frequent Itemset Mining Implementations (FIMI) Repository[2] and a Synthetic Dataset [13]. The choice of these particular data sets is not restrictive since they are widely used standard benchmark data sets, and the structure is typical of the most common application scenarios. In fact we can apply this algorithm in all scenarios provided we have the past transactions and a current customer transaction.

We compare the efficiency of ProMax algorithm against the DualRank algorithm [12], because the DualRank algorithm has already been shown to perform well when compared with the HAP algorithm and the naive approach. The DualRank algorithm [12] is the state of art in the consideration of item selection methods with customer-behavior model in the data mining literature but ProMax algorithm is based on the customer-item-behavior model.

All experiments are conducted on a 1.60GHZ Intel PC with 8GB main memory, using a Linux machine. Both algorithms were coded in C++. Profits of items are generated randomly since the way profit is generated does not affect the nature of the results generated. To perform the experiments, we took a set of four to five items which is a current customer transaction as an input and recommend top five items to the user as an output based on the past transaction history.

[2]http://fimi.cs.helsinki.fi/data

A. Performance of DualRank

Initially we have considered the performance of DualRank on the synthetic dataset. For a larger number of transactions, DualRank was not able to execute fast enough due to the huge number of computations it has to perform. Performance of DualRank is shown in the graph of Figure 1.

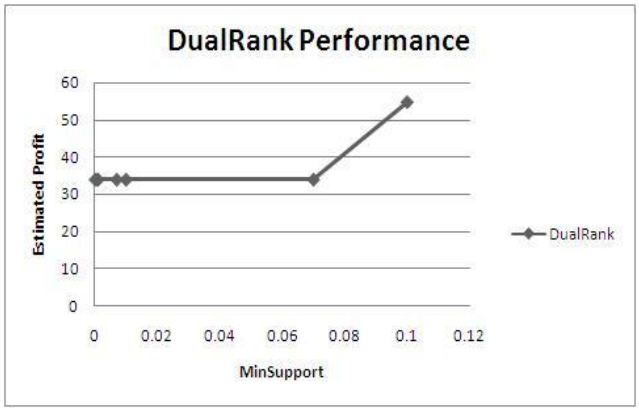

Fig. 1. DualRank Performance

For a minimum support greater than 0.1, DualRank could not be executed. The reason behind this is that, since there are very few frequent singleton items, there are no edges in the item-graph created in DualRank.

We notice that the performance of DualRank deteriorates as the min-support value is reduced. The reason is that for low minimum supports, the number of association rules generated are very large resulting in a very large matrix. This makes it difficult to perform the intensive matrix operations that DualRank requires such as the calculation of eigen values and eigen vectors. We believe the main drawback of DualRank is that its recommendations are *not* based on a particular input transaction, but are globally determined by the overall profitability of individual items.

ProMax algorithm was also evaluated under the same conditions as DualRank. Performance of ProMax is shown in the graph of Figure 3. We notice that it performs better over the entire range of min-support values that DualRank could run on. We notice that its performance increases slightly when the min-support is very low. This is only coincidental – there is as such no particular direct relationship between the min-support value and the profits generated by ProMax. This is because here the min-support can only affect the quality of clustering by modifying the intra-cluster and inter-cluster costs. Both too low and too high values of min-support could deteriorate the clustering quality. However, there is a damping affect – when min-support is reduced, there are more small items leading to a high intra-cluster cost and a low inter-cluster cost. The increase in intra-cluster cost is often compensated by the decrease in inter-cluster cost thereby resulting in no net change in cluster quality.

We have also noticed that by keeping the min-support value as constant and varying the number of items being selected, ProMax outperforms the DualRank algorithm as shown in the figure 2. DualRank always generates the static recommendations and is independent of the customer's current transaction. Hence, until the database of previous transaction history changes, DualRank always recommend the same items.

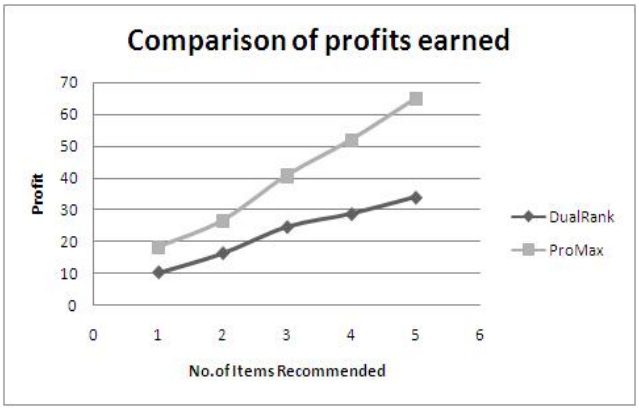

Fig. 2. Comparison of the profits earned by the algorithms based on the number of items selected

B. Performance of ProMax

In this experiment, we evaluated the performance of ProMax on both the real and synthetic datasets. The results are shown in the graph of Figure 3.

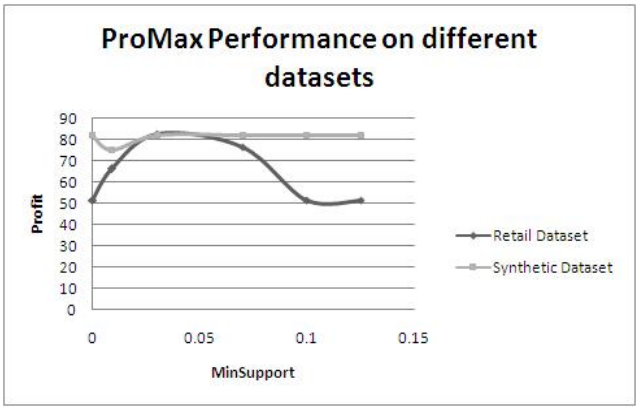

Fig. 3. ProMax Performance on Different Datasets

In this graph, the x-axis denotes the min-support, whereas the y-axis denotes the profit generated by ProMax. As per our observation, the algorithm is performing in the same manner across different datasets. For datasets which are not very sparse, profit is comparatively more. This is because of the clustering approach where the bulkiness of the clusters increases.

Also, the clustering quality effect described in the previous experiment is clearly visible in the real dataset curve. Notice that this curve has a peak at around min-support = 0.05 when clustering quality is high and deteriorates on both sides as the min-support is either increased or decreased.

VI. Conclusions and Future Work

In this paper, we presented an algorithm that combines simple ideas from clustering, recommendation systems, and the knapsack problem to recommend those items to customers that maximize the expected profit. We tested our algorithm on two popular datasets and compared our algorithm with the previously existing DualRank algorithm. Our experiments showed that our algorithm performs better than DualRank in terms of obtaining better profits, while at the same time being faster and simpler. We believe that there are few aspects to extend the current work. In the initial phase of our algorithm, clustering technique can be done more efficiently by appropriately identifying the parameters while calculating the cost. In our algorithm, item quantity was not considered but only the type of items, which can be extended as a future work.

References

[1] Brijs Tom: Retail Market Basket Analysis: A Quantitative Modelling, Ph.D dissertation, Faculty of Applied Economics, Limburg University Center,Belgium, 107–109, 2002

[2] Wang, Ke and Zhou, Senqiang and Han, Jiawei: Profit Mining: From Patterns to Actions, Proceedings of the 8th International Conference on Extending Database Technology 70–87 2002

[3] Horowitz, Ellis and Sahni, Sartaj: Fundamentals of Computer Algorithms W.H. Freeman & Company 1984

[4] Raymond Chi-Wing Wong and Ada Wai-Chee Fu: ISM Item Selection for Marketing with Cross-Selling, PAKDD 431–440, 2004

[5] Brijs, Tom and Swinnen, Gilbert and Vanhoof, Koen and Wets, Geert: Using association rules for product assortment decisions: a case study, Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, 1-58113-143-7, 254–260, 1999

[6] S Zhou and K Wang : Profit Mining, to appear in Encyclopedia of Data Warehousing and Mining, Nature, 400th Volume, 107–109, 2004

[7] Wang, Ke and Su, Ming-Yen Thomas : Item selection by ”hub-authority” profit ranking, Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, 1-58113-567-X, 652–657, 2002

[8] Brin, Sergey and Page, Lawrence : The anatomy of a large-scale hypertextual Web search engine, Comput. Netw. ISDN Syst. 30th Volume, 107–117, 1998

[9] Raymond Chi-Wing Wong and Ada Wai-Chee Fu and Wang, K. : MPIS: maximal-profit item selection with cross-selling considerations Third IEEE International Conference on Data Mining, 371-378, 2003

[10] Wong, Raymond Chi-Wing and Fu, Ada Wai-Chee and Wang, Ke : Data Mining for Inventory Item Selection with Cross-Selling Considerations, Data Min. Knowl. Discov., 11th Volume, issn 1384-5810, 81–112, 2005

[11] Wang, Ke and Xu, Chu and Liu, Bing : Clustering transactions using large items, Proceedings of the eighth international conference on Information and knowledge management, ISBN 1-58113-146-1, 483–490 1999

[12] Xiujuan Xu and Lifeng Jia and Zhe Wang and Chunguang Zhou : DualRank : A Dual-Phase Algorithm for Optimal Profit Mining in Retailing Market, ASIAN. 3818, 182–192 2005.

[13] Rakesh Agrawal : IBM Synthetic Data Generator, http://www.almaden.ibm.com/ cs/ quest/ syndata.html, 2004.

[14] Hong, Se June and Natarajan, Ramesh and Belitskaya, Ilana: A New Approach For Item Choice Recommendations, Proceedings of the Third International Conference on Data Warehousing and Knowledge Discovery, 131-140, 2001.