Recommendation system based on the clustering of frequent sets

ANDREI TOMA, RADU CONSTANTINESCU, FLOAREA NASTASE
The Economic Informatics Department
Academy of Economic Studies
Bucharest, Piata Romana n^o 62
ROMANIA
andrei.toma@ie.ase.ro, radu.constantinescu@ie.ase.ro, nastasef@ase.ro

Abstract: - Generating shopping recommendations has become a classical problem in knowledge engineering with extensive practical applications. In this article we propose a system for the generation of such recommendations based on considering both local and global influences.

Key-Words: - shopping recommendation, frequent sets, clustering, self-organizing maps

1 Introduction

In present day online business, making decisions based on analyzing the data gathered from previous periods has become not a luxury, but a necessity.
In the same trend, large online salesmen make a significant part of their revenue from the sale of products which were recommended to the customer by some kind of automated system.
Under these conditions, an entrant on the market of online commerce cannot hope to succeed without creating a relationship with its customers which will allow for repeated sales. Since an online salesman interacts with its customers mainly through its website, a very important way in which this can be achieved include efficient product recommendations.
The generation of recommendations based on stored data recording past user actions has multiple applications, from shopping recommendations to collaborative based search engines.
Since the most prevalent form in which the data comes is that of click logs or web logs, in a first step it is important to be able to extract behavioral characteristics from what is generally non structured information. A possibility is to consider as characteristics of the user the response he/she has at what can be perceived as different external stimuli. For example, different types of users can be identified if one represents the data in such a way. Some users will respond to polls and constantly view topics/products while not actually buying anything. Differentiated treatment of users with such heterogeneous objectives would lead to increasing their satisfaction with the service which integrates the recommendation engine.
In what the actual recommendation is concerned, both global and local influences are to be considered. By global influences we understand behavioral patterns which are evident at the level of the entire population of users while by local influences we understand the similarities that a particular user might have with a particular class of other users (local influences are geared more toward differentiation).
While this approach can contribute to the increase of the customer base, it is the first approach that is more likely to increase revenue in the immediate future. Although the objective of the recommendations depends on the actual seller’s strategy, this paper details the case where the recommendations are based on the preferences of users who buy frequent items.

2 Problem Formulation

Shopping recommendations can be generated either by taking into account some form of customer feedback (such as asking various questions about their preferences and building recommendations based on the answers) or in a collaborative manner, by building a model for customer preferences based on their connections with the actions of other customers.
Since the customers are generally reluctant to answer a large number of questions or, indeed, any number of questions, the second approach should be preferred.
Sales in a web based shop are generally distributed in the form of a “long tail” distribution there are of course two possible recommendation strategies, inspired from different marketing strategies. A recommendation system can either aim for the average customer (and recommend products that he/she would be likely to buy in the future) or, as a different strategy, can attempt to access the customers interested in less than frequent products (thus tapping the “tail” part of the distribution). The second strategy consists in recommending infrequently bought products to their most likely buyer even though the likelihood of an actual purchase might be small.

2.1 Available data

If recommendations are to be constructed without the explicit feedback of the customers, the data will most usually come from the logs the shop keeps of the customers' actions.

Since the recommendations will most likely be added to an existent systems and thus need to use existent data, the most general form in which the data exists is a click log form (the existent system memorizes all the clicks of all the users through the site). The size of such a log can be prohibitively large so various optimization concerns arise.

2.2 Objectives

The objective of the recommendation system is attempting to produce an accurate system based on the analysis of the frequent actions users take.

The accuracy of such a system is not necessarily a set standard as one must balance accuracy with the actual technical requirements of the system in the form in which it would be actually implemented in a production environment (which, if anything, must remain within the realm of the possible).

As such, there is a set of problems which somebody interested in implementing the system must consider.

First of all, frequent set mining can prove extremely time consuming on large volumes of data and must be optimized as much as possible. It is possible that such and optimization will prove insufficient for the requirements of a real system and that certain trade-offs between accuracy and speed must be considered.

Another point in which optimization must be considered is the clustering step, especially when using a high complexity method such as self organizing maps.

In order to obtain a working system, implementation must take into consideration the solution making some of the components of the system run offline while keeping the others online. By offline we understand running the components periodically, not at any modification of the underlying data.

Components which could be run offline in the event that the volume of data makes it necessary are the frequent pattern mining and clustering. This is acceptable since the data they produce reflect global influences which are unlikely to change on a short term. Significant modifications would have to occur in order for this data to have a negative effect on the recommendations.

The actual recommendation part of the system will be run online, as it is based on the preexistent data built through frequent pattern mining and clustering.

3 Problem Solution

The system works on a series of steps which will be presented in subsequent sections.

Some of the steps refer to algorithms which can be found in other papers and will only be referenced briefly as their presentation is outside the scope of this paper.

The idea is to generate recommendations based on two things. First we need to maintain information about each user in the form of the sets of actions that he/she performed as compared to the general population of users. These frequent sets of actions have to be separated in action classes such as buying, viewing or reaction to polls. These classes will be considered to represent characteristics of the users and thus used as a basis for the comparison between any two particular users.

Secondly we must cluster the users based on the aforementioned characteristics. As the characteristics are based on the general behavior in the population, they reflect general tendencies. We are interested to reflect the similarities within the groups of similar users, not the entire population and thus we must group the users according to similarity based on these groups that the recommendations will be produced.

Our approach is based on the clustering of feature vectors via self-organizing maps. These feature vectors are characteristics of the user defined as behavioral classesp[11].

By characteristic we understand sets of actions that the user has performed which have proven to be relevant when compared to the general population of users. These sets of actions will be obtained through frequent pattern mining, using the “a priori” method.

This is of course not the only approach possible. On one hand, the system could be modified in one of its parts, such as replacing “a priori” mining with FP-trees or another method of frequent pattern mining.

In the same line of thought, clustering could be done by another method. This could prove advantageous from a complexity point of view, as long as a method of supplying the number of clusters as an input in the system is also provided[12].

A number of different general approaches can also be applied, such as using genetic algorithms for recommendation generation or using models of the human immune system[13][14][15][16].

3.1 Gathering data

A click log might be represented in the system as in table 1 below. It will contain additional information such as location of the user and company specific data. Some of this information will be ignored in the recommendation process, while other can be used in order to obtain more accurate data (such as using the referrer to ascertain which of the open pages the user actually read).

User	Page	…	Referrer	Action
13321	311	…	x.asp	action_buy
32113	33	…	y.asp	action_view
32131	dd	…	y.asp	action_buy

Table 1

Since the information comes from user click logs, on one hand it is not separated by the criteria mentioned above (it is not separated based on the characteristics the analyst decided to take into account). On the other hand, a fair amount of pre-processing must be done to eliminate useless information. For example, especially when analyzing user view data, one cannot the conclusion that a user clicking the product has actually viewed it. The fact that multiple windows were open simultaneously for example should be reflected in pre processing (in this particular case, the window which is considered to have been viewed should be the one with a subsequent action performed in a reasonable amount of time).
Another important aspect is that if the click logs contain elements which are irrelevant for the recommendation system, these items have to be pruned out of the data. Such an example is buy data which is not actually connected to real products, such as discounts or vouchers. The representation of such entities is, of course, dependent on the implementation of the underlying system.
After the elimination of possible irrelevant information, the data used to apply the algorithm will come in the form of several entities (i.e. tables) each containing actions from a particular class. These tables correspond to the actions which are considered relevant for recommendations, which are not necessarily all possible actions. For example, the click logs will most likely store the user's access to his buy history or the help pages but these contain little to no useful information in what product recommendation is concerned.
As such, after the data gathering step we will, for example, have a table containing buy data (products bought by the users), one containing view data (products viewed by the users), one containing special offer data (the special offers the users actually clicked and read) etc.
A special mention here is that each table corresponding to an action has to be pruned again according to the specifics of that particular action. For example, the user might open multiple windows after a search, but will obviously not read all of them simultaneously. In order to make the data more relevant, only the page which has a follow-up action (a subsequent click on an element of the page) will be considered read. Buy data, in turn, should not contain all the products that were added to the shopping cart, but products which have resulted in and actual payment; products added to the cart can be, however, used to refine view data.
It is impossible to present exhaustively all the rules that should be applied in order to obtain more accurate data, as these are dependent on the underlying system.

3.2 Finding patterns with the “a priori” algorithm

For the identification of frequent patterns, one of the possible solutions is the a priori algorithm[4][8][9].
While fairly resource intensive, a fair number of optimizations can be done to ignore part of the intermediary sets which will not lead to useful results. For more details about the optimization of frequent set mining, see [1].
Also for a more in-depth description of the a priori frequent pattern mining algorithm, see [2].
One initial choice is if to apply “a priori” to the data as it was extracted from the click log or to transform it in order to allow for a vertical approach. Representing the data in vertical form would have the advantage of easier calculation of frequencies but would in turn necessitate transforming the input tables.
In the system described by the present paper, a priori mining will be done via a horizontal approach which does not require doing further transformations on the gathered data.
“A priori” is an algorithm proposed by R. Agrawal and R. Arikant which is built around the idea of using prior knowledge of lower order frequent set properties to generate higher order sets. By the “order” of a frequent set we understand the number of elements it contains.
The most important property of frequent sets is that any subset of a frequent set must also be frequent (a property which is called the “a priori” property). As such, a candidate set can be eliminated from the process if it contains at least an infrequent subset.
The frequency or infrequency of a set is defined in relation to a minimal frequency which is a parameter of the algorithm and which must be chosen so as not to eliminate any relevant sets while at the same time avoiding the inclusion of irrelevant sets.
The basis of the “a priori” property is the following line of reasoning. If an item does not satisfy the frequency condition, then any set to which it is added cannot possibly have a frequency higher than the item and thus will, in turn, not satisfy the condition either.
The basic a priori algorithm is described in the following figure (Fig. 1).

Algorithm:
P1.Go through one of the tables constructed in the data gathering phase and construct the frequences of all items. In principle, the frequencies are constructed in memory but should this be a performance constraint, they will be stored in a temporary table.
P2. Keep only the items for which the frequency is larger than a certain limit.
The frequency threshold should be chosen according to the existent data in such a way as to balance the information loss and the time needed to perform subsequent calculations.
P3. Construct candidate sets of length *i+1* by combining itemsets of length *i*.
P4. Discard constructed itemsets that fall under the frequency limit. A set is also infrequent it it contains an infrequent set of any order.
P5. Continue generating n element sets based on *n-1* element sets until no new frequent sets can be found.

Fig.1 Frequent pattern mining

As shown in figure one, the algorithm is basically organized in two major steps, one which involves generating candidate frequent sets and one which eliminates the sets which are infrequent.
At the beginning of the application of “a priori” the generation step will refer to individual items, whose frequencies will have to be generated by effectively counting all item instances.
After all items which do not satisfy the frequency condition have been eliminated, the next step is to generate 2nd order item sets by combining the remaining items. This is the generation step which will be repeated throughout the algorithm.
The next step is to verify if the generated sets are actually frequent. While the obvious method is to count all the instances of the generated sets, this will produce a lot of computational overhead and might not be feasible with a high amount of data. In order to alleviate this problem the algorithm uses the “a priori” property to eliminate possible infrequent subsets.
The next step is to continue the set generation, producing 3rd order sets by merging 2nd order sets. After the generation step, the “a priori” property is applied in a similar manner as on the 2nd level sets in order to eliminate sets containing infrequent subsets.
As a general rule, in order to find frequent sets of order k, the “a priori” algorithm will generate them by merging sets of order k-1. On this pool of candidate sets, it will then apply the “a priori” property in order to eliminate the infrequent ones.
As an optimization, sets of n elements can be generated based not only on *n-1* element sets, but also *n-2* element sets or *n-3* element sets if such sets have *count (n-k) +count (k) < count (n-1) +count (1)* where *count(x)* is the frequency of the set of *x* level based on the property that all subsets of a frequent set must be frequent themselves
The algorithm can be further optimized by exploiting a series of properties that the frequent sets have. See for example [1].

3.2.1 Recommendations based on frequent sets only
While frequent sets do not generate sufficient information to permit effective recommendations, they are an integral component in the recommendation engine.
The weakness of recommending items based on frequent sets only is that such a system will not consider the similarities between users.
The preferences of all users will have similar chance of affecting the recommendations although it can be argued that the existence of similar sets of items that users find interesting is a basis for similarity.
Recommendations can be constructed based on the frequent sets associated to a user. The mechanism we propose is presented in the following figure.

Algorithm:
P1. Consider the sets extracted for a particular user *U* and a particular characteristic *C* which form $S=(s_1,s_2,...,s_n)$
P2. Find all sets s_k belonging to the same characteristic *C* of any other user with the property that at least one element s_i of *S* is a subset of s_k (s_k contains all the elements in s_i and at least one more element)
P3. For every superset, extract the set of elements *M* that are not contained in the subset.
P4. Eliminate from *M* the actions that were already performed by the user.

Fig.2 Recommendations based on frequent patterns

The algorithm in figure 2 can also be optimized if performance constraints demand it. The number of supersets taken into calculations can be adjusted by considering supersets only for the sets of maximum size (*n*) or including supersets for sets of lower size (*n-k* with a strictly positive *k*).

While the algorithm described in figure 2 is relatively effective in producing items which are likely to be interesting to the user, it does so only by estimating effects on a global level. The results should be narrowed down by taking into account the preferences of the users which are behaviorally similar. Furthermore, even for a particular user, the algorithm takes into account only the actions taken from a particular class of possible actions. For example, it can give a viewing recommendation, but it ignores which products the user has actually bought.

Optimization of the result would first involve introducing a way to consider the whole of user behavior and not just the current category. As such recommending similar products to a user currently viewing a product must involve not only the products the user previously viewed, but also, for example, the fact that his reaction was one of interest to a promotion involving certain products.

Secondly, it must be considered that, taking into account the whole of the user's behavior, there might be greater similarity between him/her and a certain set of users. Should such a similarity exist, those users' behavior should be the basis for recommendation.

3.3 Clustering users using self-organizing maps

In order to find out which users are similar in behavior, one can apply as series of algorithms such as K-NN or SOM (which is the one we have opted for). The advantage of using self-organizing maps for clustering users according to preference stems from the fact that there is no initial need to know the number of clusters. With SOM, the number of clusters itself is produced by the algorithm without the necessity of giving the as an input[3][5][6][10].

However, since SOM is a "heavier" algorithm than some of the alternatives, in the sense that the needed computer time is much bigger, one could presumably run SOM just to detect the number of clusters and then use it as a basis for the implementation of a "lighter" algorithm.

Should such an approach prove necessary, the algorithm used for clustering in the actual system should probably be a variation of naïve Bayes, which would produce accurate results with an acceptable loss in accuracy.

The basic algorithm for SOM based clustering is presented below, in figure 3. For a more complete description see [3][7].

Algorithm

P1. Define the size of the network (the number of input variables, the size of the external layer), the learning rate.

P2. Calculate the time constant, the maximum radius of the network.

P3. Allocate random values to the intensities of the connections to the external layer.

P4. Select an user.

P5. For the selected the user, calculate the neuron on the external layer which is closest (winning neuron). The distance between an input set and the external neurons is calculated via a version of edit distance.

P5. Calculate the radius of the winning neuron's neighbourhood and adjust the intensities for the neurons in the neighbourhood with decreasing influence from the centre to the borders.

P6. As long as there are more users, go back to step **P4**. With passing iterations of the algorithm, the learning rate and neighbourhood radius are affected by a degeneration function.

P7. Group the users according to which neurons on the external layer fire when the input layer is activated with the user's characteristics.

Fig.3 SOM clustering

The concept of self-organizing maps was introduced by Teuvo Kohonnen and is one of the more popular clustering mechanisms[3].

Self-organizing maps are neural networks which allow grouping of the data based on the relations existent in the data itself. Their training is unsupervised, without the need for external input under the form of training sets.

The characteristics taken into account when classifying the users are vectors of frequent sets.

Each characteristic is a set of frequent actions taken in a class of actions. Such a class might be buying or viewing.

As such, a user will be represented as a vector of characteristics such as $U = (C_1,..,C_n)$, where the number of characteristics n is the number of entities constructed in the data gathering step. C_1 might thus represent buying, while C_2 might represent viewing. The degree of similarity of a user U_i to a user U_j will be represented by the sum of the similarities between each of the corresponding characteristics of the two users. The use of the distance between two users is presented in figure 4 with $d(x,y)$ the distance function between two users

defined as the sum of the distances between each of the characteristics of the users.
Each of the characteristics will be on the form of a set vector, with the sets present in a characteristics being frequent sets mined through the “a priori” algorithm. A characteristic C will thus have the form $C = (s_1,..,s_n)$, where s_i is a frequent set.
Distances between two characteristics, which are needed to calculate the distance between users are calculated based on the appearance of the same frequent sets in both characteristics, as shown in figure 5. Sets contained in only one characteristic which have a corresponding superset in the other are considered relevant based on the reasoning that in the future they might generate common sets.
Clustering through self-organizing maps consists of mapping the data from the n-dimensional space of the input vectors, which in our case represent the users, to a 2-dimensional space. This of course has the added advantage of being able to represent similarities within the data in a human readable form. There are of course possible optimizations to the representation of the data which may reveal further correlations (such as representing the outer layer on a torus shape or making it 3-dimensional).
The first step in applying the algorithm to the data is determining the structure of the network. Also, a series of parameters needed in the training of the network have to be determined at this stage.
It must be mentioned that the results on different structures of the network applied to the experimental data should guide the analyst in determining the best solution to be selected.
A self organizing map is composed from an input layer with a number of nodes equal to the number of components of the input vectors and an n-dimensional output where n is the number of dimensions of the map on which we want to represent the input vectors.
All input nodes are connected to all outer layer nodes while there are no connections between input nodes or output nodes. The intensities of the connections between the input and output nodes will be used for the classification of the users after the training process.
In our case the number of input nodes will be equal to the number of relevant characteristics identified in the system. The nodes will thus correspond to the different actions the users might take (buying, viewing etc.).
The output layer serves as a map to the similarities between the input vectors. Inputs which are more similar will appear closer on the map.
The dimensions of the output map have to be selected such as to avoid losing classification accuracy while keeping the map at a sufficiently low size. Higher size maps need more resources in order to classify the inputs so the size must be selected that manages to balance accuracy with resource expenditure.
We opted for an initial outer layer dimension of 10x10 which proved large enough to allow for separation between the clusters of users. Considering that the outer layer is relatively small, the needed calculations were well within the accepted limits.
Other parameters which must be decided at this stage are the learning rate and the initial radius of the starting neighborhood.
The learning rate controls the speed at which the network adapts to changes in the training data and should be selected so as to control the network’s response to extreme values.
The initial neighborhood radius will be set to the maximum possible (the dimension of one side of the outer layer) and then be degenerated as the algorithm progresses.
The next step is to initialize the weights of the connections between the input nodes and the outer layer nodes. The weights are initialized with small random values ($0<w<1$) for all the connections from input nodes to output nodes.
After the initial preparations, input vectors are selected and presented to the network. Since the system must adapt while online input vectors will be presented in the order in which modifications occur. As a note, this process will take place offline, that is the training algorithm will be run periodically starting from the already determined weights and using as training vectors the newly added/modified user vectors.
As such, the initialization of the weights with small random values will only take place on the first run of the training algorithm.
After a vector is selected, the next step is to find which output neuron is the winning neuron in relation to the input vector.
The custom distance function used is described in figure 4 and figure 5.

$$dist(x, y) = \sum d(x_i, y_i)^2$$

Fig.4 Distance between users

The index i counts the characteristics of a particular user. The function d is the actual distance function between characteristics, where n is the number of common sets the characteristics have, m is the number of subsets characteristic x_i contains for the sets in y_i and k is the number of subsets characteristic y_i contains for the sets in x_i.

$$d(x_i, y_i) = n + 0.5*(m + k)$$

Fig.5 Distance between characteristics

The distance function is justified by the fact that similarities are reflected both by the existence of identical behavioral patterns for two users but also by the possibility of future similarity. The quotient *0.5* used as a weight for probable future similarity should actually be determined through experimentation on the data.
After the winning neuron has been found the weights of the connections from the input neurons to the winning neuron are corrected, as well as the weights of the neurons in the neighborhood of the winning neuron.
Since the network is supposed to stabilize in time, the radius of the neighborhood is decreased by multiplying it with a degeneration function.
Also, after each iteration, the learning rate should also be decreased.
The network is retrained continuously (iterative approach) by supplying it with new inputs corresponding to user actions that were not already presented to it.
This means that when a new user appears, the corresponding vector will be trained into the network. Also, when a user's behavior extends, the corresponding vector will also be retrained into the network.

3.3.1 Recommendations based on clustering and frequent sets

Since we now have a method of grouping the users according to their actions in certain categories, we will use the information to refine the algorithm presented in figure 2.
The revised algorithm, which takes into account both the global and the local influences, is presented below in figure 6.

Algorithm:
P1. Consider the sets extracted for a particular user *U* and a particular characteristic *C* which form $S=(s_1,s_2,...,s_n)$
P2. Find all users which belong to the same cluster with *U*
P3. Find all sets s_k belonging to the same characteristic *C* of any other user found in P2 with the property that at least one element s_i of *S* is a subset of s_k (s_k contains all the elements in s_i and at least one more element)
P4. For every superset, extract the set of elements *M* that are not contained in the subset.
P5. Eliminate from *M* the actions that were already performed by the user.

Fig.6 Recommendation algorithm

Since searching for the users in the same cluster can generate performance problems and the clusters have a high degree of stability, the current cluster to which an user is supposed to belong is stored and the inclusion of users in clusters is recalculated periodically.
The algorithm produces a "recommendation basket" from which items can be selected, either on an objective basis, such as the items that appear as a result of an increased level of similarity (higher order supersets) or by intersecting the set with a set of items the system including such a model has an interest in recommending (e.g. top selling products).

3.4 Practical concerns

Depending on the size of the data, some of the operations necessary for recommendation generation might be more usefully done on a periodical basis (offline), while the recommendation system itself would function "live" (online).
For example, the discovery of frequent patterns could be done offline. Since they contain a minimal frequency level parameter they allow for accuracy preservation even in an offline scenario.
In the same way, the inclusion of users in particular clusters can be done offline, as long as the system provisions for new users.
Various performance concerns should be addressed if they affect the system, as the development would ideally be dynamic that is to say if a certain degree of optimization is needed (such as partitioning the data) then it should be applied[1]. However, without a practical need, optimizations should not be implemented if they lead to a loss in accuracy.
Two schemes hypothetical systems are presented in figure 7 and 8 below.
The system in figure 7 is the one presented in section 3.2.1, while the system in figure 8 is the improved one.

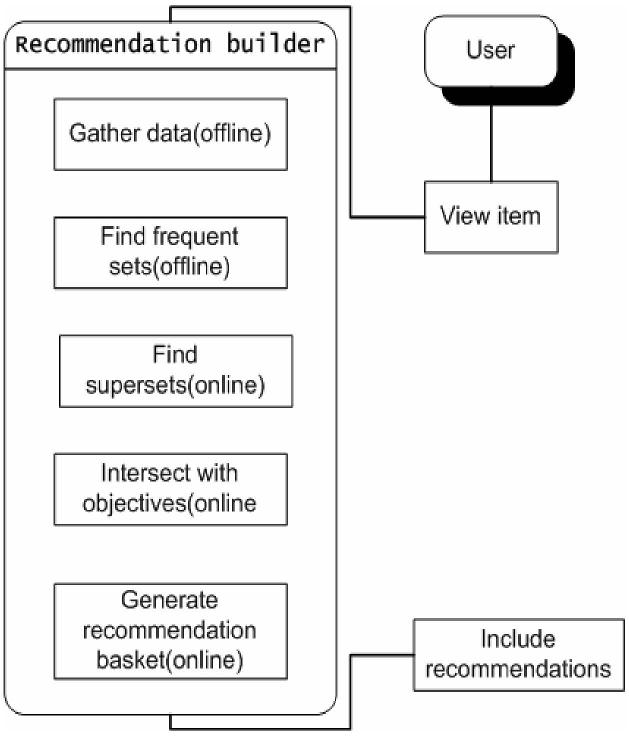

Fig.7 Flow of the recommendation system with clustering

As discussed above, the simpler system has a number of downsides, stemming mainly from the fact that it does not take into account all the characteristics at once. It also does not consider the correlations of the current user to the other users.

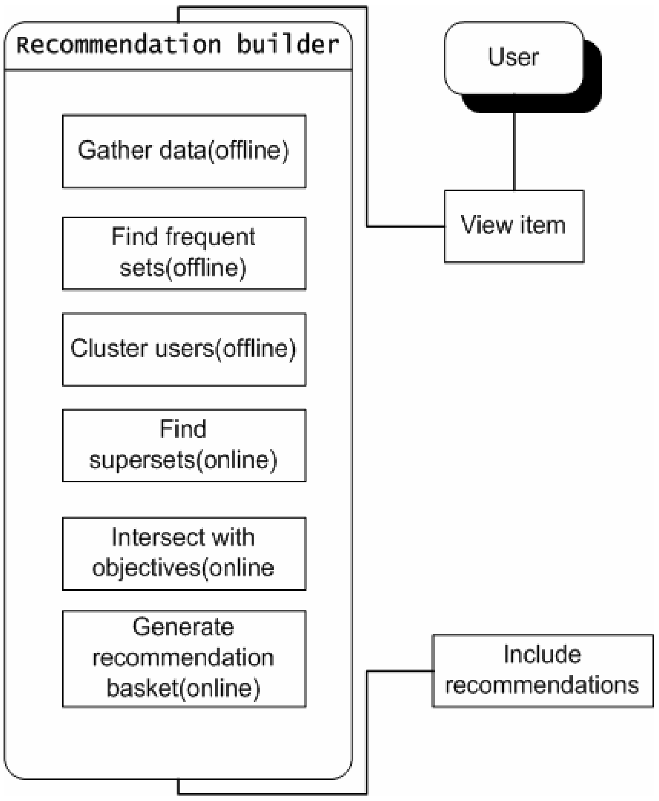

Fig.8 Flow of the recommendation system with clustering

As shown above in figure 8, the system has two parts, an offline and online one. The operations marked offline will only be performed if a large enough interval has passed. When a user makes a request, this interval is checked and, if necessary, the offline tasks are started in the background.
Whenever a user accesses a page corresponding to a product he/she will be presented with product recommendations.
While recommendations are made to the user, the system checks if the offline operations have been performed. Regardless if the offline operations have been started in the background the system then finds the users which are located in the same cluster. Parsing the sets of the selected users, the system then selects those which are supersets of any set associated to the user for which recommendations are currently constructed.
From the selected supersets, the elements which do not appear in any of the sets of the user for which the recommendations are constructed are selected. This is the recommendation basket which can then be used according to the marketing objectives of the company using the system.
For example, the recommendation basket can be intersected with the list of the most popular products. The result of the intersection can then be recommended to the user.
In what the offline part of the system is concerned, if at a particular request it is decided to update the data, data gathering is started. The tables needed to run frequent set mining are reconstructed/updated.
After the data gathering step, frequent sets are mined via the “a priori” algorithm. The next step is to cluster the users. Considering the large volume of data, the structure of the trained network will be kept and the new and modified users will be presented to the network in the form of supplemental training data.

4 Conclusion

There are two possible approaches to recommendation, one based on direct user feedback, such as through polls, one based on collaborative construction of the recommendations.
There are distinct advantages in choosing the latter since it does not rely on, even limited, user sincerity and since the data, if less informative, is more accurate.
However, extracting different analyzable characteristics can lead to interesting results.
Recommendation would avoid considering the data as homogeneous (such as in the case of using simply page id’s as a basis for recommendation). The advantages stem from the fact that while actions taken by users,

while on a basic level represent accessing certain items, can also be considered as pointing to more advanced behavioral patterns.

The solution of constructing characteristics from by considering user actions as referring to different categories is an intermediary approach, between explicit responses to questions, in which case, of course the users could give inaccurate answers, and simple access of items.

Using association rules as a basis for recommendation is an attempt to generalize past behavior of users in order to predict future behavior, or more exactly possible candidates for future behavior.

Limiting the search to users which are significantly similar to each other increases the likelihood of accurate prediction. In order to obtain such a limitation, user clustering must be implemented. While this operation can be done by a multitude of means, self-organizing maps have the advantage of discovering not only the clusters to which the users belong but also the cluster number itself. However, there are alternatives so self-organizing maps which are less resource intensive, providing the analyst supplies another means of determining the number of clusters (which would then become an input of the system).

If the number of clusters is known or the volume of data is too large to apply any other method, naïve Bayes is to be considered, as the least demanding clustering method in terms of necessary computations.

As of this moment the system has not been implemented; the results of the eventual implementation will be published in a future paper at the time that they become available.

A possible extension of the system comes from a different commercial approach. As it stands, the system tries to recommend the items which have the highest likelihood of being bought next. However, an alternate approach would be to try to approach the problem of increasing sales by tapping into the tail of the distribution.

Such an approach would try to increase the volume of sales not by appealing to the regular customers but trying to attract the people who would not regularly buy at the online shop.

Online shopping has an advantage in trying to attract customers interested in uncommon items due to the lower logistic costs compared to traditional stores.

In present, people make most of their current purchases online so the volume of sales compared to traditional stores is constantly increasing.

Under these conditions, an entrant on the market of online commerce cannot hope to succeed without creating a relationship with its customers which will allow for repeated sales. Since an online salesman interacts with its customers mainly through its website, a very important way in which this can be achieved include efficient product recommendations.

The generation of recommendations based on stored data recording past user actions has multiple applications, from shopping recommendations to collaborative based search engines.

Generating recommendations based on frequent behavioral patterns also has the advantage of being easily adapted to user profiling. User profiles would then be used not only for decisions regarding the online part of the business, but also for the salesman's public relations decisions.

References:

[1] Bart Goethals, Efficient Frequent Pattern Mining – Ph.D. Thesis, Universiteit Limburg, pp.21-29, 2002

[2] Jiawei Han, Micheline Kamber, Data mining concepts and techniques 2nd edition, Morgan Kaufmann Publishing pp.234-239, 2005

[3] Teuvo Kohonen, Self-Organizing Maps 3rd edition, Springer Publishing, 2000

[4] Ian H. Witten, Eibe Frank, Data Mining: Practical Machine Learning Tools and Techniques, 2nd Edition

[5] Angel F. Kuri-Morales , Automatic Clustering with Self-Organizing Maps and Genetic Algorithms II: an Improved Approach, Proceedings of the 5th WSEAS International Conference on Neural Networks and Applications, Udine, 2004

[6] Angel F. Kuri-Morales, Automatic Clustering with Self-Organizing Maps and Genetic Algorithms", Recent Advances in Simulation, Computational Methods and Soft Computing, WSEAS Press, 2002

[7] Francesco Maiorana, Performance improvements of a Kohonen self organizing classification algorithm on sparse data sets, Proceedings of the 4th WSEAS/IASME International Conference on Educational Technologies, Corfu, 2008

[8] Agrawal, R., Imielinski, T., Swami, A. Mining associations between sets of items in large databases, Proceedings of ACM SIGMOD International Conference on Management of Data, Washinton D.C., 1993

[9] Agrawal, R., Srikant, R. Fast Algorithms for mining association rules in large databases. Proceedings of 20th International Conference on Very Large Databases, Santiago de Chile, 1994

[10] Charalampos Vassiliou, Dimitris Stamoulis Drakoulis Martakos, A Recommender System Framework combining Neural Networks & Collaborative Filtering Proceedings of the 5th WSEAS International Conference on Instrumentation, Measurement, Circuits and Systems, Hangzhou, 2006

[11] Xuejun Zhang, John Edwards, Jenny Harding, Personalised online sales using web usage data mining, Computers in Industry 58, Elsevier, 2007
[12] Susanne Still, William Bialek, How Many Clusters? An Information-Theoretic Perspective, Neural Computation, Volume 16, Issue 12 2004
[13] Zhiyong Zhang ,Olfa Nasraoui Mining search engine query logs for social filtering-based query recommendation, Applied Soft Computing ,Volume 8, Issue 4, Elsevier September 2008
[14] Olfa Nasraoui, Fabio A. González,Cesar Cardona,Carlos Rojas,Dipankar Dasgupta: A Scalable Artificial Immune System Model for Dynamic Unsupervised Learning, GECCO 2003
[15] Olfa Nasraoui,Raghu Krishnapuram Robust Multi-Resolution Web Usage Mining with Genetic Niche Clustering (2008)
[16] Wei-Po Lee, Chih-Hung Liu, Providing Personalized Information Services by Developing Intelligent Recommender Systems GA