Target Advertisement based on Cohesive Structure in a Social Network

Umasankar Das
Silicon Institute of Technology
Bhubaneswar, Odisha

Girija Prasad Mohapatra
Silicon Institute of Technology
Bhubaneswar, Odisha

Kunal Shrivastava
Indian Institute of Technology,
Guwahati

ABSTRACT
In this paper, we propose a model that utilizes the concept of social networking for target advertising of various products. This approach examines the cohesive subgraph structure within a network and model a recommendation system for target advertising. We test the proposed model using synthetic data and our model can help the advertising company to advertise the products at those nodes only, where the user have the interest in that product. This will effectively reduce the advertisement cost.

Keywords
SNA, NN, ANN, Cardinality, cohesive, Social network; Targeted advertising; Recommender system; Knowledge discovery

1. INTRODUCTION
The motivation for the developing this model comes from the problem of selecting the user of the social networking site to whom the product company will publish the target advertisement, so that they can publish their advertisement in smaller search space[1][6] and that to be with optimum cost and time. Actually when we visit any social networking site, we usually come across the advertisements published there. But most of the time the published advertisements are of no use for user. So those advertisements go unused. The advertising company usually spend a lot of money in publishing advertisement to each of the node of a network, it will be beneficial if we can advertise the products at those nodes only, where the user have the interest in that product. This will effectively reduce the advertisement cost.

The effectiveness of targeting range of product to a set of people has long been recognized by the target advertisers, for two main reasons. First, the amount of product/service information available to customers is increasing like anything, and hence it is desirable to help customers to search through the information to find the product/service they want[3][5]. Second, understanding the needs of current and potential customers is an essential part of customer-relationship management. The ability to accurately and efficiently identify the needs of customers and subsequently advertise products/services that they will find desirable will increase customer-retention, growth, and profitability of a business.

2. RELATED WORK
Most recommendation algorithms start by finding a set of customers whom purchase and rate items. The algorithm aggregates items from these similar customers, eliminates items the user has already purchased or rated, and recommend items to the user. Two popular versions of these algorithms are collaborative filtering and cluster models. Other algorithms including search based methods and our own item to item collaborative filtering focus on finding similar items, not similar customers. For each of the user's purchased and rated items, the algorithm attempts to find similar items. It then aggregates the similar items and recommends them.

2.1 Traditional Collaborative Filtering
A traditional collaborative filtering algorithm represents a customer as an *N*-dimensional vector of items, where *N* is the number of distinct catalog items. The components of the vector are positive for purchased or positively rated items and negative for negatively rated items. To compensate for best selling items, the algorithm typically multiplies the vector components by the inverse frequency (the inverse of the number of customers who have purchased or rated the item), making less well-known items much more relevant. For almost all customers, this vector is extremely sparse. The algorithm generates recommendations based on a few customers whom interest is most similar to the user. It can measure the similarity of two customers ,A and B, in various ways; a common method is to measure the cosine of the angle between the two vectors .The algorithm can select recommendations from the similar customers' items using various methods as well, a common technique is to rank each item according to how many similar customers purchased it

To use collaborative filtering to generate recommendations is computationally expensive. It is $O(MN)$ in the worst case, where M is the number of customers and N is the number of product catalog items, since it examines M customers and upto N items for each customer. However, because the average customer vector is extremely sparse, the algorithm's performance tends to be closer to $O(M + N)$. Scanning every customer is approximately $O(M)$, not $O(MN)$, because almost all customer vectors contain a small number of items, regardless of the size of the catalog. But there are few customers who have purchased or rated a significant percentage of the catalog, requiring $O(N)$ processing time. Thus, the final performance of the algorithm is approximately $O(M + N)$. Even so, for very large data sets such as 10 million or more customers and 1 million or more catalog items. the algorithm encounters severe performance and scaling issues. It is possible to partially address these scaling issues by dimensionality reduction techniques such as clustering and principal component analysis can reduce M or N by a large factor. Unfortunately, all these methods also reduce recommendation quality in several ways. first, if the algorithm examines only a small customer sample, the selected customers will be less similar to the user. second, item space partitioning restricts recommendations to a specific product or subject area. third, if the algorithm discards the most popular or unpopular items, they will never appear as

recommendations, and customers who have purchased only those items will not get recommendations. It is also possible to reduce the number of items examined by a small, constant factor by partitioning the item space based on product category or subject classification.

Dimensionality reduction techniques applied to the item space tend to have the same effect by eliminating low frequency items. Dimensionality reduction applied to the customer space effectively group's similar customers into clusters; as we now describe, such clustering can also degrade recommendation quality.

2.2. Cluster Models

To find customers who are similar to the user, cluster models divide the customer base into many segments and treat the task as a classification problem. The algorithm's goal is to assign the user to the segment containing the most similar customers. It then uses the purchases and ratings of the customers in the segment to generate recommendations. The segments typically are created using a clustering or other unsupervised learning algorithm, although some applications use manually determined segments. Using a similarity metric, a clustering algorithm groups the most similar customers together to form clusters or segments. Because optimal clustering over large data sets is impractical, most applications use various forms of greedy cluster generation.

These algorithms typically start with an initial set of segments, which often contain one randomly selected customer each. They then repeatedly match customers to the existing segments, usually with some provision for creating new or merging existing segments. For very large data sets especially those with high dimensionality sampling or dimensionality reduction is also necessary. Once the algorithm generates the segments, it computes the user's similarity to vectors that summarize each segment, then chooses the segment with the strongest similarity[7] and classifies the user accordingly. Some algorithms classify users into multiple segments and describe the strength of each relationship. Cluster models have better online scalability and performance than collaborative filtering because they compare the user to a controlled number of segments rather than the entire similarity

2.3 Search-Based Methods

Search or content based methods treat the recommendations problem as a search for related items. Given the user's purchased and rated items, the algorithm constructs a search query to find other popular items by the same author, artist, or director, or with similar keywords or subjects. If a customer buys the Godfather DVD Collection, for example, the system might recommend other crime drama titles, other titles starring Marlon Brando, or other movies directed by Francis Ford Coppola. If the user has few purchases or ratings, search based recommendation algorithms scale and performs well. For users with thousands of purchases, however, it's impractical to base a query on all the items. The algorithm must use a subset or summary of the data, reducing quality. In all cases, recommendation quality is relatively poor. The recommendations are often either too general (such as best-selling drama DVD titles) or too narrow (such as all books by the same author). Recommendations should help a customer find and discover new, relevant, and interesting items. Popular items by the same author or in the same subject category fail to achieve this goal.

2.4 Item-to-Item Collaborative Filtering

Amazon.com uses recommendations as a targeted marketing tool in many email campaigns and on most of its Web sites' pages, including the high traffic Amazon.com [2] homepage. Clicking on the "Your Recommendations" link leads customers to an area where they can filter their recommendations by product line and subject area, rate the recommended products, rate their previous purchases, and see why items are recommended.

The feature is similar to the impulse items in a supermarket checkout line, but our impulse items are targeted to each customer.Amazon.com extensively uses recommendation algorithms to personalize its Web site to each customer's interests. Because existing recommendation algorithms cannot scale to Amazon.com's tens of millions of customers and products, we developed our own. Our algorithm, item-to-item collaborative filtering, scales to massive data sets and produces high-quality recommendations in real time [3].

3. PROPOSED METHOD

3.1 The procedure is as follows:

1. Maintenance of user's input for different attributes in matrices (0-1) and files.
2. Applying string encoding algorithm to these inputs to generate a numeric pattern.
3. Training Neural Network.
4. Deciding decision factor (i.e degree of interconnections and priority decisions).
5. Application of the trained Neural Network to sort out the subgroup from the network and publishing advertisement based on degree of interconnections and priority shown in fig. 1.

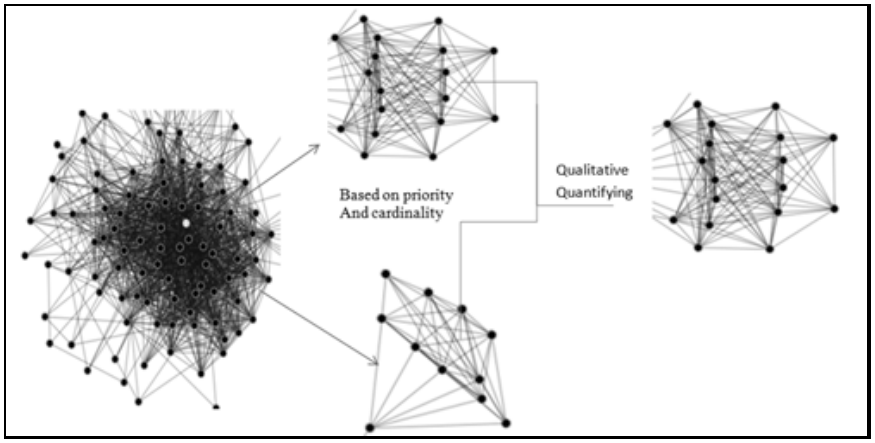

Fig 1: Cohesive Subgroups Model

3.2 Maintenance of user's input for different attributes in matrices (0-1) and files.

In our database we will maintain a tables (or matrices) for network connection and tables for each attribute**.** In network matrix there will be equal number of rows and columns both equal to the number of users in the network. In the network matrix n(i,j) ,each entry n(i,j) will be equal to 1 if ith user is connected to the jth user otherwise entry will be equal to 0.

The attribute table would be in matrix form where each row will correspond to an individual and each column defines an instance of a particular attribute. So if there are 'm' numbers of users then there will be 'm' number of rows in each and every matrix. So a matrix of an attribute of size mxn means that there are m numbers of persons in the network and there are n numbers of instances of that attribute.

A table called instance table is also maintained along with the matrices for each category in which there will be two fields, one for the instance name and another for the corresponding column number values assigned to the instance in the attribute matrix.

At beginning we will create the matrices of concerned attributes with 3 or 4 instances defined and all the value of the matrix set to '0' ,for example we will create food matrix with instances chocolates, noodles, biscuits etc and each entry a(i , j)=0 . If the user's entry under food category matches with the any of the existing instances then the cell value corresponding to that particular user and instance is assigned a value '1'. The instance table is also initialized with the predefined instance value and column number corresponding to that particular instance.

In case if the user entry does not matches with any of the existing instances then a new column will be created dynamically in the corresponding attribute matrix and the new column will be assigned to that particular entry. Values of the column will be set to 0 and 1 depending on the user's entry. The instance table will also be updated at the same time for the new entry.

In a similar manner matrices and instance table of all the concerned attributes are maintained and will be used later in the descriptive stages of the advertisement.

3.2.1Why to create columns for the instance dynamically:

It is not possible that all the instances of a particular attribute are already known. The entry in the attribute field totally depends on the user and as there will be large number of users so there will be large number of variations. Predefining all the instances is not possible so every time when a new instance will be encountered a column will be created and will be assigned to that particular instance. The instance table will be updated accordingly. Dynamic creation of column and assigning that to a particular instance will broaden our field of advertisement and will avoid the unwanted entry from the database.

3.3 Applying string encoding algorithm to these inputs to generate a numeric pattern.

The instance table that is maintained regularly is used for finding a cohesive group in the network to which advertisement will be published. Classification is done by neural network which is defined earlier. The name of the instances of a particular attribute will be inputted to the neural network and used for classification purpose. As neural network deals only with the numeric values so it cannot input string (instance name) to the neural network, so it has to convert these string values into numeric values. We will use the string encoding algorithm defined above for string encoding. An instance matrix will be maintained for all the concerned attributes which will be a two dimensional matrix in which the numeric values of the instance name returned by the string encoding algorithm will be stored along with the column number assigned to the instance in the attribute matrix. It is just like the instance table but here in place of name of the instance there will be the numeric values of the instances.

The string encoding algorithm which will be used should be fast and do not have any redundancy problem i.e produces same code for two different string values. For string encoding we are going to use 26 alphabets (A-Z), 10 decimal digits(0-

9) and some special character (!,@,#,$,&,*), so there are a total of 42 characters.
Now we have divided these 42 characters into 7 sets where each set will contain 6 characters. The set will be as like:

Set.1: A B C D E F
Set.2: G H I J K L
Set.3: M N O P Q R

Set.4: S T U V W X
Set.5: Y Z 0 1 2 3
Set.6: 4 5 6 7 8 9
Set.7: ! @ # $ & *

For encoding the string is processed from left to right, for a particular character its set number and its position in that particular set is stored in two different arrays. When the whole string is processed both the arrays are checked for their size, if the array size is less than the assigned value (assigned 15) then 0 is appended in the array to make its size 15. After this both the arrays are concatenated to make an array of size 30. Now these arrays of integer will be the numeric values of the string which will be inputted to the neural network for further processing. The size of the numeric value is fixed to a particular value because the number of input nodes in the input layer is fixed and cannot be changed and as we can have string of different lengths so a value is set which will define the length of the numeric values. This algorithm is not case sensitive. Example showing how coding is done: suppose we have a string "hello". First two arrays will be created one for the set number and one for the position of character in that particular set. Suppose the arrays are array1 and array2 • First character is 'h'. h is in 2 set and position of 'h' inside the set is 2, these values will be added to the corresponding arrays.

Array1: 2

Array2 : 2

• Second character is 'e'. 'e' is in 1 set and position of 'e' inside the set is 5,

now these values will be added to the corresponding arrays.

Array1: 2 1

Array2: 2 5

• Third character is 'l'. 'l' is in 2 set and position of 'l' inside the set is 6,

now these values will be added to the corresponding arrays.

Array1: 2 1 2

Array2: 2 5 6

• Fourth character is 'l'. 'l' is in 2 set and position of 'e' inside the set is 6,

now these values will be added to the corresponding arrays.

Array1: 2 1 2 2

Array2: 2 5 6 6

• Fifth character is 'o'. 'o' is in 3 set and position of 'o' inside the set is 3,

now these values will be added to the corresponding arrays.

Array1: 2 1 2 2 3

Array2: 2 5 6 6 3

Size of both the array is 5 which is less than 15 so 10 zeros are appended in

both the array and the arrays are concatenated together to form a large array of size 30.

So the numeric code for the string "hello" is 2 1 2 2 3 0 0 0 0 0 0 0 0 0 2 5 6 6 3 0 0 0 0 0 0 0 0 0 0

In this manner every string is encoded. It is not necessary that the size of the numeric value returned should always be 30. This value can be changed and depends on the programmer. But it must be taken care that the size of the numeric values decide the number of nodes of input layer in the network, so size cannot be changed in between the program rather it is set initially to a certain value and then neural network is defined accordingly

As our string encoding algorithm return numeric values of size 30 so there will be 31 column in the instance matrices, column 1-30 for the numeric values and 31st for the corresponding column number values in the attribute matrix. So if an instance matrix has size mxn then there will be 'm' number of instances of particular attribute and n will be constant having value 31. So an instance matrix can have size mx31.

3.4 Training Neural Network

In the proposed algorithm we are using artificial neural network to sort out the interested persons or groups. The neural network which we are using has a three layered structure,

1. **Input layer** : it consists of 30 input nodes and one bias node
2. **Hidden layer**: it consist of 10 nodes and one bias node
3. **Output layer**: it consists of a single node. Bias is not added to this layer.

Back propagation algorithm is used for training the neural network.The ANN which we will use consists of 30 input nodes, 10 hidden nodes and 1 output node. We will use sigmoidal function as activation function. The activation function is

$$F(x) = \frac{1}{1+e^{-x}}$$

Input layer nodes do not have any activation function in it. The input to input layer node is directly forwarded to the output of the node. But the input to hidden and output layer are first processed with the activation function and then forwarded to the output of the node.

There are weight matrices between two consecutive layers. We denote by Wij the weight of the connection from unit xi of input layer to unit yj of hidden layer. We denote by Vij the weight of the connection from unit vi of hidden layer to unit zj of output layer. The output from one layer is forwarded to next layer after evaluation with these weight matrices.

The equation for feed forward processing are:

Input to input layer, II= X (given)

Output of input layer, OI= X

Input to hidden layer, IH= $\sum_{i=1}^{m} OI.W$,where

m=number of nodes in input layer

Output of hidden layer, OH=F(IH)

Input to output layer, IO= $\sum_{j=1}^{n} OH.V$, where

n=number of nodes in hidden layer

Output of output layer, O= F(IO)

Learning

For learning and training we will use backpropagation algorithm. In this approach we calculate error at each stage (iteration) of training and from that error; weight vectors between hidden and output layer and between input and hidden layer are adjusted so that after a certain number of iteration the output of the ANN will approximate the desired output.

Error calculation

If the network output is O= {O_1, O2, O3,….O_N} and target output is T={T_1,T_2,T_3,….T_N}, then error at each node of output layer

$\in_i = 1/2(T_i-O_i)^2$

Total error for one training pattern, E= $\sum_{i=1}^{n} \in_i$

Total error for n number of samples $E^T=1/2 \sum E$

Training:

$[W]^{new}=[W]^{old} + [\Delta W]$

$[V]^{new} =[V]^{old} + [\Delta V]$

$\Delta W=-\eta \frac{\delta E_T}{\delta W}$

$\Delta V=-\eta \frac{\delta E_T}{\delta V}$

Where ŋ=learning rate and its values varies between 0-1. It is preferable to take the value above 0.9 for fast and graceful training.

The neural network will take numeric values of any instance as the input and will give the column number assigned to that particular instance in the attribute matrix as the output but before using the neural network we have to train it based on which it will give the correct result. We have to train our neural network for all the concerned attributes and for training purpose we have to first create a training pattern. So we have to create training pattern for every attributes and here we are using instance matrix as the training pattern. For a particular attributes let's say that there are m number of instances then there will be m row in the instance matrix , one for each instance and the column number will be equal to 31. For a particular attribute the entire instance matrix is inputted to the network, first 30 column values will be taken as the input to the neural network and the 31st value will be the corresponding output of that particular input. One by one each row is evaluated and error is calculated. Based on the errors the weight between the layers are modified and adjusted in an iterative manner. The network is trained for entire matrix and for each row, and when the network gets trained i.e the error of the network comes below 0.01% the weight matrices (that is the weight between the input and hidden layer and between hidden and output layer) are returned and are stored. In a similar manner the weight matrices for each and every attributes are calculated and are stored in the weight matrix which will be used later when the neural network is used for classification.

3.5 Calculating degree of interconnection and priority decision.

When an instance is inputted to the trained neural network it gives column number corresponding to that instance as the output. Cells having value 1 in this column will be searched and corresponding users are grouped together. As there will be a large number of users so advertisement cannot be published to all of them but it will be published to a certain group of people. The decision factor which determines which users to be selected among all is the degree of interconnection each user and its priority or influence factor.

Degree of interconnection is calculated by finding the number of friends of that particular user. This can be calculated by row addition of the network matrix. After addition a mx1 matrix will be created, where m =number of users in the network and each value will give the number of friends of that users.

Priority decision is made by calculating number of times the profile of a user is being visited. An mx1 matrix is also maintained for this where each cell value corresponds to the number of times the profile of mth person is being visited.

From the selected group of people the person with highest degree and high priority will be selected accordingly and advertisement will be published to them. The number of persons to which advertisement will be published totally depends on Advertisement Company that to how many persons they want to advertise.

3.5.1 Why to take 'number of friends' and 'number of times profile is visited' as the decision factor:

Number of friends defines the cardinality of a person in the network. Person having more number of friends will have more interconnections in the network hence have high degree of cardinality in the network. Number of times a person profile is visited defines the influence factor of that person in the network. More the number of times a person profile is visited by different people show that the person has much more influence on the network and advertisement published to a person will be indirectly published to all the persons who are visiting that person profile.

3.6 Application of the trained Neural Network along with the decision factor to sort out the subgroup from the network.

When an instance is inputted to the trained neural network it gives column number corresponding to that instance as the output. Cells having value 1 in this column will be searched and corresponding users are grouped together. As there will be a large number of users so advertisement cannot be published to all of them but it will be published to a certain

group of people. The decision factor, which determines which users to be selected among all, is the degree of interconnections of each user and its priority or influence factor. For degree of interconnection we will cross check that how many times that profile is being visited by different users. For this we will sort the user according to the number of friend and the influence factor, the person at the top of the array will be preferred for advertisement. Number of person to be advertised depends on the advertisement company and also minimum number of person will be selected to whom if advertisement is published will be visible to each and every person in the network either directly or indirectly.

3.7 How Advertisement Done

The advertisement company is provided an interface in which they will select the category and instance group to which their product belong, the interface is shown below in fig. 2

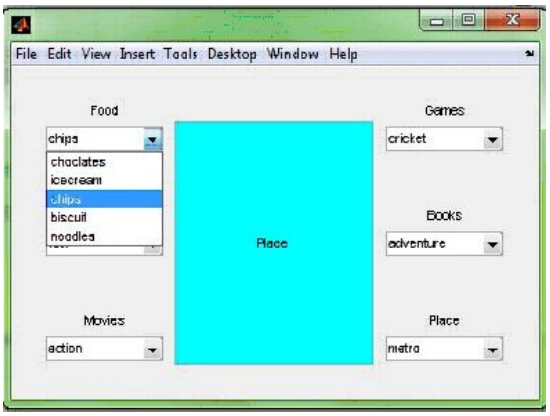

Fig 2: Interface for advertisement

If suppose the advertisement company has to add for a chips product then it will select chips from the food category and place its advertisement in the advertisement section. When enter is pressed, the group which is selected (here it is the 'chips') is encoded and changed into numeric values. These numeric values are inputted to the trained neural network. The neural network will load its weight vector from the food weight matrix stored earlier. The inputted numeric value is processed over this weight vector to produce the column number value as the output. Now from the whole food matrix only the column number which is the output of the neural network is searched for the interested person. Persons corresponding to cell of this particular column having value equal to 1 are grouped together and a cohesive group is selected based on the decision factors and advertisement is published to all these people.

The output for the above advertisement is as follows:

column_num = , list_on_priority =

Columns 1 through 10

2100 2100 943 1357 1357 4960 3090 4991 2010 3714

list_on_cardanility =

Columns 1 through 10

1000 3621 3621 2010700 629 3375 3898 3898

Advertisemnet will be published to node number:

Columns 1 through 9

2100 2100 943 1357 1357 4960 3090 4991 2010

Columns 10 through 18

3714 1000 3621 3621 2010700 629 3375

Columns 19 through 20

3898 3898.

4. CONCLUSION & FUTURE WORK

In our proposed model we are dynamically adding instances to the instance matrix by creating a column in the matrix whenever a new instance name is entered by the user. But sometimes it may be possible that the new instance entered will be related to the previously existing instances and in this case if new column is created then redundancy will increase in the database. To avoid this redundancy we use data decomposition algorithm to group all the related instances together and assign one column for all of them. If some user leaves the network then a discontinuity will be created in our network graph and because of this discontinuity a group of people will get disconnected from the network and will not be taken under consideration by our advertisement algorithm. So to deal with this an algorithm will be developed which will forecast the node which is going to leave the network and when that node leaves the network the algorithm will maintain continuity in the network by connecting the underlying node with the main network.

5. REFERENCES

[1] George Prassas, Katherine C. Pramataris, Olga Papaemmanouil, Georgios J. Doukidis "A Recommender System for Online Shopping Based on Past Customer Behaviour" 14th Bled Electronic Commerce Conference Bled, Slovenia, June 25 - 26, 2001.

[2] Greg Linden, Brent Smith, and Jeremy York,"Amazon.com Recommendations Item-to-Item Collaborative Filtering" Published by the IEEE Computer Society,February-2003.

[3] Wan-Shiou Yang , Jia-Ben Dia , "Discovering cohesive subgroups from social networks for targeted advertising" published by Department of Information management, Carlos Castro-Herrera, ChuanDuan , Jane Cleland-Huang, Bamshad Mo basher,"A Recommender System for Requirements Elicitation in Large-Scale Software Projects" published at Systems and Requirements Engineering Center , Center for Web Intelligence , DePaul University,March-2004

[4] National Changhua University of Education, September-2004.

[5] Armstrong, G., & Kotler, P. "Marketing: An introduction. Upper Saddle River", NJ: Prentice-Hall-1999.

[6] Dewan R., Jing B.,and Seidmann A."Informative Narrowcasting with Consumer Search",Proceedings of the 35th Hawaii International Conference on System Sciences - 2002.

[7] Tan D., Liew S., Tan T.,Yeoh W.,"A Feature Selection Model for Binary Classification of Imbalanced Data Based on Preference for Target Instances",4th Conference on Data Mining and Optimization (DMO)-2012.