5-2002

Mining relationship graphs for effective business objectives

Kok-Leong ONG

Ee Peng LIM
Singapore Management University, eplim@smu.edu.sg

Wee-Keong NG

DOI: https://doi.org/10.1007/3-540-47887-6_56

Mining Relationship Graphs for Effective Business Objectives[⋆]

Kok-Leong Ong, Wee-Keong Ng, and Ee-Peng Lim

Centre for Advanced Information Systems, Nanyang Technological University,
Nanyang Avenue, N4-B3C-14, Singapore 639798, SINGAPORE
ongkl@pmail.ntu.edu.sg

Abstract. Modern organization has two types of customer profiles: active and passive. Active customers contribute to the business goals of an organization, while passive customers are potential candidates that can be converted to active ones. Existing KDD techniques focused mainly on past data generated by active customers. The insights discovered apply well to active ones but may scale poorly with passive customers. This is because there is no attempt to generate know-how to convert passive customers into active ones. We propose an algorithm to discover relationship graphs using both types of profile. Using relationship graphs, an organization can be more effective in realizing its goals.

1 Introduction

The modern organization has two types of customer profiles: active and passive. We define an *active customer* as one that contributes to the business objectives of the organization. For example, an active customer may engage in a purchase [4], surfed an organization's Website [2,3], or signed up for trial/free products and services. A passive customer is thus one that has no contributions other than its profile. In the early era of KDD, passive customers do not really exist. However, the ubiquity of e-Commerce and data acquisition technologies has made such information readily available. Essentially, passive customers are acquired by means of third party information providers, partnerships with other organizations, or registration of trial products and services.

This presents an important issue. We have, in fact, been looking at data generated by active customers, and the insights obtained are better catered towards them. For example, the knowledge used in the bundling of items in a supermarket [1] is likely to work well for active customers. However, the same know-how may not reach a passive customer since they do not contribute any data for KDD. More importantly, the knowledge obtained does not provide any indication about how passive customers may be converted into active ones. With the belief that there are insights in data that can help convert passive customers, we propose the mining of relationship graphs (or simply graphs in this paper) from both types of customer profiles. In a graph, nodes represent "customers"

[⋆] This work was supported by SingAREN under project M48020004.

and edges represent "relationships". By finding relationships between active and passive customers, we can capitalize on the active customer's ability to influence the passive party to make a contribution to the organization's goal.

Example: The insurance industry in Singapore is a very competitive market. Ad-hoc approaches such as selling on the street or calling a party in the house are attempts that is difficult and ineffective. Hence, agents depend largely on their immediate family members, friends and relatives, and in the later stage, the relationships from the current pool of people. To help their agents reach beyond relatives and friends, an insurance company teamed with an information provider to allow its brokers to query a particular existing customer (i.e., active) and have the graph of the active customer shown. From the graph, a broker may select a related passive customer via the active customer under its portfolio, and thus, path an opportunity to a potential customer.

Certainly, there are other possibilities. The focus is the use of relationships between two people to create a business opportunity. In Asia, the use of relationships is sometimes more effective in achieving business objectives then plain marketing campaigns. This is how relationship graphs differ from other techniques – they find insights in both types of customer profiles that help organizations target potential customers effectively. Formally, the mining of relationship graphs involves finding the set of passive customers in a database that have the *strongest* relationship with a given active customer. In the next section, we give a formal definition of the problem and its parameters. Section 3 presents the algorithm for mining relationship graphs. We then conclude our discussion in Section 4.

2 Problem Formulation

Let $C = \{c_1, c_2, \ldots, c_i\}$ be the set of all passive and active customers in the organization and $R = \{r_1, r_2, \ldots, r_j\}$ be the set of all possible relationships that exist between two customers. A tuple t in the database D is of the form $\langle c_\alpha, c_\beta, r\rangle$ where $c_\alpha, c_\beta \in C$ and $r \in R$, that defines the existence of a relationship r between c_α and c_β. A **relationship graph (G)** is a set of tuples $\{t_1, t_2, \ldots, t_k\} \subset D$ that represents a set of relationship paths from the active customer c_a to a set of passive customers $\{c_{p_1}, c_{p_2}, \ldots, c_{p_j}\}$. A **relationship path** is a set of tuples $P = \{t_x, t_y, \ldots, t_z\} \subset G$ where $t_1.\alpha = c_a \vee t_1.\beta = c_a$ and $t_k.\alpha = c_{p_i} \vee t_k.\beta = c_{p_i}$ such that $\forall t_x \in P - \{t_1, t_k\}, \exists t_y \in G,\ t_x.\beta = t_y.\alpha$.

For simplicity and compactness of the database, all relationships are undirected. Even with this simplification, the potential relationships possible from an organization warehouse is huge[1]. To limit the number of relationship paths in G, only those strongest in the goal of the organization's objectives are considered

[1] A domain expert may choose to relate two entities by their address, interest, working organization, profession, geographic region, expertise etc., and this generates huge number of tuples.

for a given c_a. The **minimum strength** (δ) is the score that a relationship path must achieve in order to be selected into the graph. This is a quantitative value that act as a measure on the likely effectiveness of a relationship path in the view point of the domain expert. This is computed by a user defined function $\mathcal{S}(\{t_0, t_1, \ldots, t_k\})$ where the strength is determined by the evaluation of the relationships $t_0.r_0$, $t_1.r_1$, $\ldots$, $t_k.r_k$. The motivation behind $\mathcal{S}$ is to model complex interactions between different relationships that cannot be represented by the direct assignment of weights used in typical graph exploration algorithms (e.g., shortest path [5]).

We also define the distance to determine the "radius" of the graph G with the c_a as the "centriod". Intuitively, as the distance gets larger, the effectiveness of the relationship weakens. Therefore, it may not be cost effective to mine beyond a certain distance. Formally, the **distance** (π) is the maximum allowable distance for a relationship path from the active customer to a passive customer. Together with the notion of strength, we obtain the **size** (φ) of a relationship graph. The size is thus the maximum number of allowable relationship paths that satisfy δ and are within π. Therefore, φ represents the top φ strongest relationship paths.

3 Mining Relationship Graphs

Figure 1 shows the algorithm for mining a single relationship graph given an active customer. This is done in two steps. First, tuples that are involved in the current computation are identified and selected into the respective ordered list. Once the lists are constructed, the next step is to scan the lists to construct the top φ strongest path for a given active customer.

Upon entry to the algorithm, π ordered list are created (line 3) and represented as $b_0, b_1, \ldots, b_{\pi-1}$ where each holds tuples selected from D. Starting with b_0, we select all tuples in D that contain the active customer c_a. Instead of the lexicographical ordering in the database, we order the tuples such that the first customer (i.e., $t.\alpha$) is c_a using the function "ArrangeTuple" in line 4. We then compute the strength using $\mathcal{S}$ and then sort, in descending order, the strength of their relationship (line 5). This generates the relationship graph with a distance of 1. The loop from lines 6-11 constructs the remaining list up to the distance specified by π. Notice the construction of the k^{th} list is dependent on the results of the $(k-1)^{th}$ list. This is equivalent to a breadth-first search of tuples in D to construct the "super-graph" with c_a as the "centroid" up to the distance of π. In line 7, we identify the selection criteria for the next list and stores it in B_c. We then select tuples (line 8) containing a customer in B_c. Line 11 prune tuples containing all active customers. Such tuples are unnecessary by the definition and should be removed to minimize memory consumption and faster computation.

The second half of the algorithm finds the subgraph of c_a that satisfies φ and δ. Since all tuples in b_0 contains c_a, all tuples in this group must be considered regardless of the relationship strength. For each tuple in b_0 (line 12), we construct the path up to the maximum distance possible by finding matching tuples to form a relationship path. In the algorithm, T holds the set of tuples

```
01 procedure MineGraph
02 begin
03     let $B = \{ b_0, b_1, \ldots, b_{\pi-1}\}$ and $G = \varnothing$ and $B_c = \{ c_a \}$
04     let $b_0$ = ArrangeTuple($\{ t \in D \mid t.\alpha = c_a \vee t.\beta = c_a \}, B_c$)

05     foreach $t \in b_0$ do $t.\delta = \mathcal{S}(\{t\})$; Sort $b_0$ using $\delta$ in descending order

06     for ($k = 1; k < \pi; k{+}{+}$) begin
07         let $B_c = B_c \cup \{ t.\beta \mid t \in b_{k-1} \}$
08         let $b_k$ = ArrangeTuple($\{ t \in D - \{b_0, b_1, \ldots, b_{k-1}\} \mid t.\alpha \in B_c \vee t.\beta \in B_c \}, B_c$)

09         foreach $t \in b_k$ do $t.\delta = \mathcal{S}(\{t\})$; Sort $b_k$ using $\delta$ in descending order
10     endfor

11     $b_k = b_k - \{ t \in b_k \mid t.\alpha$ is active customer $\wedge\, t.\beta$ is active customer$\}$

12     foreach tuple $t \in b_0$ do
13         let $T = \{ t \}$ and $t_c = t$
14         for ($k = 1; k < \pi; k{+}{+}$) begin
15             if ( FindTuple($t_c, b_k$) $= \varnothing$) then exit for-loop else $T = T \cup \{ t_c \}$
16         endfor

17         if ($\mathcal{S}(T) \geq \delta \wedge |T| \leqslant \pi$) then
18             $G = G \cup \{ T \}; \varphi = \varphi - 1$
19             if ($\varphi = 0$) then end by outputting solution G
20         endif
21     endfor
22 end

23 procedure ArrangeTuple(list of tuples $T$, $B_c$)
24 begin
25     foreach tuple $t \in T$ do
26         if ($t.\alpha \notin B_c$) then Swap contents of $t.\alpha$ and $t.\beta$
27     endfor
28     return $T$
39 end

30 procedure FindTuple(tuple $t_c$, ordered list $b_k$)
31 begin
32     foreach tuple $t \in b_k$ do
33         if ($t.\alpha = t_c.\beta$) then return $t' \mid t'.\alpha = t.\beta \wedge t'.\beta = t.\alpha$
34         if ($t.\alpha = t_c.\alpha$) then return $t$
35     endfor
36     return $\varnothing$
37 end
```

Fig. 1. Algorithm for mining relationship graph of a given active customer.

that forms a relationship path (line 13) and t_c is the tuple under consideration. The function “FindTuple” attempts to find a corresponding tuple that joins to the current tuple t_c and thus, extends the distance of the current path by 1 in the next bin b_k (line 14-16). Once the path is obtained, we measure its strength (line 17). If it satisfies δ, it is added to the solution G and φ is decremented (line 18-19). This repeats for all tuples or until the top φ solution is discovered. The algorithm then terminates by printing the solution G in line 19.

3.1 Design Heuristics

In most cases, approximately φ tuples of b_0 need to be scanned if φ is smaller than the number of strong relationships. For the remaining lists, an average of

$\varphi \times log_2(|b_k|)$ tuples are scanned if a binary search is assumed. This reduction in the scan is based on a simple heuristic that allows the relationship graph to be discovered quickly. Specifically, tuples in the list are sorted by their relationship strength. For a given path P, we observe that if $\mathcal{S}(P) \geq \delta$, then adding the next tuple t_c, regardless of its strength, is likely to be stronger than both T and t_c being weak. As such, sorting each list translates to a higher probability of finding strong paths without traversing every tuples in the lists.

In the best case (assuming a binary search), only $\frac{1}{h} \times (\varphi + log_2(|b| \times (\pi - 1))$ tuples need to be scanned, where h is the probability of finding a matching tuple such that the length of the current relationship path is extended by 1 and b is the average number of tuples in each ordered list. Compare this to the trivial approach in which $b_0 \times b_1 \times \ldots \times b_{\pi-1}$ (i.e., $|b|^{\pi}$) tuples are scanned and its relationship strength computed to find the top φ list. Clearly, the use of such heuristics is desirable when the relationship database and the number of active customers to discover in each run is large.

3.2 Scoring Relationships

Earlier, we mentioned that $\mathcal{S}$ is user-defined. To illustrate how $\mathcal{S}$ may be defined, let R be the set of all relationships in the relationship database. We partition R into $R_1, R_2, \ldots, R_j$ such that $R_1 \cup R_2 \cup \ldots \cup R_j = R$ and $R_1 \cap R_2 \cap \ldots \cap R_j = \varnothing$. For each $R_k \subset R$, $0 \leqslant k \leqslant j$, we define a function $f_{R_k}(T)$ as

$$f_{R_k}(T) = \begin{cases} \exists t \in T, t.r \in R_k & v_k \\ otherwise & 0 \end{cases} \tag{1}$$

where v_k is a real value that is assigned to each group of relationships that are considered to be of the same level of importance. Then, a possible score for any relationship may be defined as

$$\mathcal{S}(T) = C + f_{R_1}(T) + f_{R_2}(T) - f_{R_3}(T) \times \frac{1}{f_{R_4}(T)} + \ldots \times f_{R_j}(T) \tag{2}$$

In the above, the default score is given by C. Note that one possible interpretation of the above model is that normal relationships add score to $\mathcal{S}$ and relationships that are weak in the goal of the organization objective subtract scores from $\mathcal{S}$. In addition, high influence relationships contribute by multiplying the score while very negative relationships can be modelled by reducing the score through division. Most important of all, as T grows, the value of $\mathcal{S}$ changes. This allows a relationship discovered later to override a preceding relationship and hence modify the overall strength of the path.

4 Summary

We have introduced the problem of mining relationship graphs using active and passive customers profiles. For a given active customer, we are interested in finding a subgraph containing strong relationship paths that reach passive customers.

Since a relationship graph is essentially an undirected graph [6], our intuition is to look at existing graph exploration algorithms. However, algorithms such as the "shortest path" [5] returns a single solution between two vertices and designates that as the optimum. In the case of mining relationship graphs, there can be more than one solution between two vertices as long as the solution satisfies δ. In addition, there can be no input from the domain analyst and there is no provision for freedom in the solution that is necessary to model relationships. Furthermore, the costing function assumes that the cost of an edge cannot overwrite any preceding edges traversed. In the case of relationship graphs, certain relationships may have a high influence over another preceding relationship that it is worth the effort (i.e., by computation of $\mathcal{S}$) to take an alternative solution.

The work reported in this paper is the result of a discussion with a company looking into visualizing its customer database for effective channelling of their business efforts. The results of visualization is overly complex and motivated the use of algorithmic methods to discover appropriate relationships in the customer database. The algorithm is currently implemented as part of a trial to assess its effectiveness over plain visualization.

Although the paper is set in the context of an active and passive customer, we would like to point out that the graph is equally applicable in other situations. For example, two active customers may purchase the same software. Using the graph, the software company may use the influence ability of one who upgraded to a newer version to create an upgrade on the other. This achieves the business objective of the organization as well. It is thus important to note that the application of the relationship graph depends largely on the goals and nature of the organization.

References

1. Agrawal, R. and Srikant, R.: Fast Algorithm for Mining Association Rules. Proc. 20th Int. Conf. on Very Large Databases, Santiago, Chile, Sep. 1994.
2. Cooley, R., Mobasher, B. and Srivastava, J.: Web Mining – Information and Pattern Discovery on the World Wide Web. Proc. 9th IEEE Int. Conf. on Tools with Artificial Intelligence, Nov. 1997.
3. Dua, S., Cho, E. and Iyengar, S. S.: Discovery of Web Frequent Patterns and User Characteristics from Web Access Logs – A Framework for Dynamic Web Personalization. Proc. Int. Conf. on Application-Specific Systems and Software Engineering Technology, pp. 3-8, 2000.
4. R. D. Lawrence, G. S. Almasi, V. Kotlyar, M. S. Viveros and S. S. Duri.:Personalization of Supermarket Product Recommendations. Applications of Data Mining to e-Commerce – a Special Issue of the Int. J. on Data Mining and Knowledge Discovery, 2001.
5. Swamy, M. N. S. and Thulasiraman, K.: Graphs, Networks, and Algorithms. John Wiley & Sons, Inc., 1981.
6. Wilson, R. J. and Wakins, J. J.: Graphs – An Introductory Approach. John Wiley & Sons, Inc., 1990.