BeTracker: A System for Finding Behavioral Patterns from Contextual Sensor and Social Data

Hsun-Ping Hsieh, Cheng-Te Li, Shou-De Lin
Graduate Institute of Networking and Multimedia, National Taiwan University, Taipei, Taiwan
sandoh714@gmail.com, reliefli@gmail.com,sdlin@csie.ntu.edu.tw

***Abstract*—In this work, we integrate the contextual information provided from sensor data and the social relationships collected from online social networks to construct a system, termed BeTracker. We aim to find and track the frequent and representative behaviors for any user-input individual or social structural information. We claim combining physical contacts from sensor data and virtual online interactions can reveal real-life human behaviors. In our BeTracker, we mine the temporal subgraph patterns as the discovered behaviors from sensor-social data transactions. The user-given information, which is the target to observe, can be (a) an individual (to find her daily behaviors), (b) a relational structure (e.g. linear, triangle, or star structure) (to find the frequent and contextual interactions between them), and (c) a relational structure with partially assigned individuals and sequential time periods (to observe their interactions that follow certain temporal order). In the experimental part, we demonstrate promising results of different queries and present the system efficiency of the proposed behavioural pattern mining.**

Keywords-temproal subgraph pattern; frequent pattern; query; mobile sensor data; social network

I. INTRODUCTION

Nowadays, sensors are often built-in modern mobile phones so that they have the potential to provide insights into the human dynamics in the real-life environment. For real-world individuals, in addition to equipping with mobile sensors, most of them have online identities in current social networking services, such as Facebook and Google+. It is the fact that the mobile sensors and the social networking interweave and provide the rich dynamic contexts of human interactions. Such two kinds of complement data are beneficial for us to investigate human behaviors. For example, if Bob and Mary are acquainted with each other in Facebook, we can infer that they stay together in the encounter records from mobile sensors. On the contrary, if they are strangers, it is less possible for them to have the encounter records of sensors.

In this work, we integrate the contextual information provided from sensor data and the social relationships collected from online social networks to construct a system, termed BeTracker. The objective of BeTracker system aims to find and track the frequent and representative behaviors for user-interested individual information. We intend to summarize the interaction behaviors between individuals from both the sensor encounter data and the online social connections. Given the observation target as the query, our system will find and return a kind of temporal subgraph patterns as the discovered behaviors from the sensor-social data transactions. The observation target can be (a) an individual (to find her regular behaviors), (b) a relational structure (e.g. linear, triangle, or star structure) with partially assigned individuals (to find the frequent and contextual interactions between them), and (c) a relational structure with partially assigned individuals and sequential time periods (to observe their interactions that follow such order in the period). We will show some promising results of different queries in experiments section.

Consider an encounter scenario as an example, in which Bob works with John during 8am and 5pm every day. After working, Bob eats dinner with his wife, Mary, during 6pm and 8pm. On the other hand, Tom and Ted usually go to bar together during 9pm and 11pm. In Figure 1(a), we can find that such interactions and represent their encounter records as a structure form in which their interaction durations are captured on the edges. Based on such elemental case, from the overall encounter data, we can model the human behaviors by transforming all encounters with the interaction durations into the temporal behavioral networks. In each behavioral network, each node represents an individual and an edge stands for two persons encounter within a certain time interval (or several time intervals) from sensor data. We collect and construct a series of such kind of temporal behavioral networks day by day. On the other hand, to have the real acquaintance information between individuals to help reveal human interaction behaviors, we collect their social relationships from the online virtual social networking service (i.e., Facebook). For the example in Figure 1(a), its corresponding illustrated social network is shown in Figure 1(b). Furthermore, we also elaborate an example of discovered behavior for a user-specified query. Figure 1(c) illustrates a resulting behavior for the query individual, Bob. Such behavioral structure shows that Bob have a non-friend meet with John for ten hours, and then he stay together with Mary about two hours. We call such kind of graphs as a temporal subgraph pattern which tells who are the ones that regularly interact with Bob (in nodes) and when does their interactions happen (on edge labels).

Mining such behavioral patterns has some benefits. For example, the opportunistic routing of messages in delay tolerant networks (DTN) is still a challenge problem due to the intermittent connectivity and the lack of continuous end-to-end paths between the nodes. If a system can efficiently track an individual's frequent and representative behaviors, it is possible to enable sensors to have better forwarding decisions and improve the routing efficiency. From the perspective of social networks, mining such patterns can help us define relationships more precisely than simply using the information from online social network since the patterns can be implicitly used to measure the closeness of friendships in the social network. In [1], their work detected temporal events from sensor data. However, the method cannot perform on dynamic sensor environments, even on mobile sensor data. Existing works, such as [2][3][4], focus on exploiting the social network to improve the routing efficiency on sensor systems. Comparing to their works, our mined behavioral patterns from sensor data not only captures the real-life interactions between individuals, but also enables discovering the underlying human behaviors.

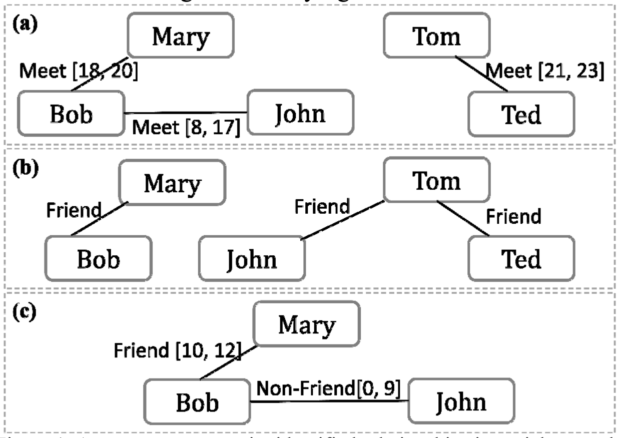
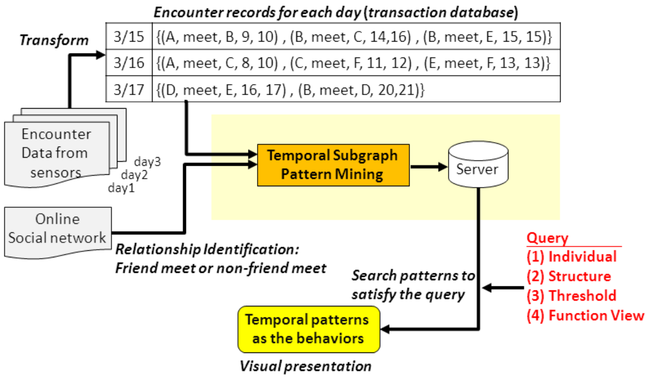


Figure 1. An encounter scenario, identified relationships in social network and an example temporal subgraph pattern

II. System Overview

Our BeTracker is constructed on a mobile sensor network, which comprising devices can detect other same devices carried by different people within a radius of 10 meters, and the corresponding social network. Figure 2 shows the framework of our system. We assume the encounter records collected from sensors for each day are uploaded through the base stations to a central database. On the other hand, we exploit the individuals' social network to identify the relationships between individuals. Combining such sensor and social contexts, we devise a the temporal subgraph pattern mining method to find the underlying temporal frequent behaviors, called temporal subgraph patterns and store them in our database. Given a query individual or structure with certain thresholds, sequential time periods, and the function view (which will be elaborated in Section 4), our system will find and return the behavioral patterns related to the query.



Encounter records for each day (transaction database)
Transform
3/15 {(A, meet, B, 9, 10) , (B, meet, C, 14,16) , (B, meet, E, 15, 15)}
3/16 {(A, meet, C, 8, 10) , (C, meet, F, 11, 12) , (E, meet, F, 13, 13)}
3/17 {(D, meet, E, 16, 17) , (B, meet, D, 20,21)}
Encounter Data from sensors
day3
day2
day1
Temporal Subgraph Pattern Mining
Server
Online Social network
Relationship Identification: Friend meet or non-friend meet
Query
(1) Individual
(2) Structure
(3) Threshold
(4) Function View
Search patterns to satisfy the query
Temporal patterns as the behaviors
Visual presentation

Figure 2. System Overview

III. Methodology

In this work, we regard the behaviors in a day as a temporal snapshot. By collecting temporal graphs in a series of days, we construct a transaction database of behavioral networks, in which each network stands for a graph recording behaviors from sensor data in a day. To easily perform our temporal-based method, we sort the edges by the associated timestamps and transform the graph into an edge sequence. For example in Figure 2, there is a behavior sequence, {(A, *meet*, B, 9, 10) (B, *meet*, C, 14, 16) (B, *meet*, E, 15, 15)} on March 15. The edges are sorted by their start times and then by their end times (Unit: hour). In each transaction (network), we then exploit the corresponding social network to provide more information about relationships identification between individuals. A relationship can be identified as "friend meet" (FM) or "non-friend meet" (NFM) by the friend-list retrieved from online social network. If two people perform friend meet, we can infer that stay together and have some interactions from the encounter records. If another two persons have non-friend meet, it may indicates that they do not acquaint with each other and just stay in the same place occasionally. After constructing networks for each day, we perform the temporal subgraph pattern mining algorithm on these networks. The temporal subgraph pattern, is defined as {(u_1, l_1, v_1, t_{s1}, t_{e1})...(u_h, l_h, v_h, t_{sh}, t_{eh})}, where t_{s1}=0, and all the edges in the pattern are sorted in increasing order. To measure the importance of a pattern, the strength of a pattern is calculated by counting its *support*, which is defined as the number of graphs containing P in the network database. A pattern P is *frequent* if its *support* is not less than *minsup*, where *minsup* is a user-specified minimum support threshold. The proposed algorithm has two stages. First, we mine all frequent patterns of length one (denotes 1-patterns) in the database. Then, for each 1-pattern, we build the projected database to help discover more patterns. For example, if we have a pattern P=(A, FM, B, 0, 1), the corresponding projected database of P in a certain day is {(B, FM, D, 1, 2) (A, NFM, D, 2, 3) (A, FM, B, 4, 6) (A, FM, C, 7, 9)}, which is the P's postfix. By scanning different projected databases from all transactions contain P, we can find a local pattern e, say, {(B, FM, D, 1, 1)}. We concatenate P and e to form a new pattern {(A, FM, B, 0, 2) (B, FM, D, 1, 1)}. The concatenations are recursively performed in a depth-first search manner until no more frequent patterns can be found.

Moreover, we adopt the closed pattern mining concept introduced by Pasquier et al. [5]. A close frequent pattern means there does not exist any super-pattern with the same frequency in database. A suitable closed pattern mining strategy can decrease number of frequent patterns during mining process so that it can improve efficiency and memory usage of the proposed method.

During the mining process, we use the closure checking and pruning strategies to reduce unnecessary candidates. The first strategy is *Same projected database removal*. If P_1

is a super-pattern of P_2 and both share the same projected database, P_2 is not needed to be grown because the patterns generate from P_2 will be not closed patterns. The second strategy is *Forward checking scheme*. A pattern P is not closed if there exists a frequent pattern e in P's projected database, whose support is equal to P's support. The third strategy is *Backward checking scheme*. A pattern P is not needed to be grown if there exists a frequent pattern e before P, whose support is equal to P's support. Thus, every pattern generated from P is contained by the pattern generated from concatenating P and e and both patterns have the same support. By applying these strategies, the closed frequent temporal subgraph patterns can efficiently mine.

The default threshold for *minsup* is 5%. Therefore, the patterns whose support is not less than 5% are stored in the database. Once a user inputs a structural query, we will return patterns by the following process: a query $\{(qu_1, ql_1, qv_1)...(qu_m, ql_m, qv_m)\}$ is contained by a pattern $\{(pu_1, pl_1, pv_1, pt_{s1}, pt_{e1})...(pu_n, pl_n, pv_n, pt_{sn}, pt_{en})\}$ if there exists a sequence of integers $j_1<j_2<\ldots<j_n$ so that $qu_i=pu_{ji}$, $ql_i=pl_{ji}$, $qv_i=pv_{ji}$, $i=1,2,...,n$. We can use this property to check query existence no matter users assign structure or sequential time periods.

The time complexity of the mining temporal subgraph algorithm is $O(n \times l \times p)$ and the space complexity o is $O(n \times l \times r)$, where n is the number of transactions in the database, l the average number of edges in all transactions in the database, p the number of frequent patterns, and r the length of the longest pattern. Here, due to the page limit, we skip the proof here.

IV. Function Views

Our BeTracker provides the following functional views.

A. Local View

The local view retrieves the behavioral patterns for a query individual. For example, if we query an individual, Bob, the corresponding pattern is shown in Figure 1(c).

B. Pair View

Given a certain pair of individuals, BeTracker will return the behavioral patterns that they stay together. Such pair-based behavioral patterns allow observing the duration (i.e., how long) and the meeting type (one-on-one or group meeting) of their encounter.

C. Structure View

BeTracker allows users to input the structural query with sequential time periods. The canonical query structures could be the star, circular, and linear topologies. And users can specify any kinds of simply structures. Moreover, we allow assigning the sequential time periods on edges, which can help understand the order and time duration of behaviors between people. In short, such structural query with sequential time periods enables users to investigate delicate human interactions. The example behavioral pattern for the query in Figure 3(a) is shown in Figure 3(b).

D. Source-Target View

Another characteristic query of our BeTracker is the source and target individuals. We report the shortest path between the given source and target, which is expected to be quite different from one day to another. The behavioral path can help the packet forwarding decision for the dynamic routing in sensor networks. Take transaction database in Figure 2 as example, the shortest path between A and B on March 15 is *A-B-E*. The shortest path between A and B on March 16 is *A-C-F-E*.

E. Global View

Finally, BeTracker can present the whole network constructed from the sensor data with the social contexts. Through such overview of the behavioral network, users can easily locate themselves or other target individuals and understand any regular, interesting, or abnormal interactions in the designated single or consecutive days. For example, the network on March 15 in Figure 2 is shown as in Figure 4.

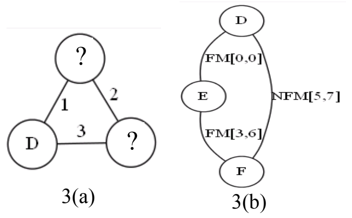


Figure 3. (a) Query (b) Pattern

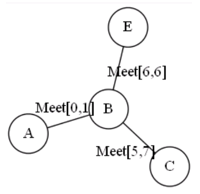


Figure 4. A global view

V. Experiments

In this section, we show the experiments on our system efficiency. Our BeTracker utilizes the CRAWDAD data [6], which is a wireless network data consisting of sensor mote encounter records and the corresponding social network data of a group of participants at University of St Andrews. This data contains 27 invent devices amongst 22 undergraduate students, 3 postgraduate students and 2 members of staff. Participants were asked to carry the devices whenever possible over a period of 79 days. We use the participants' Facebook friend lists to as the social contexts.

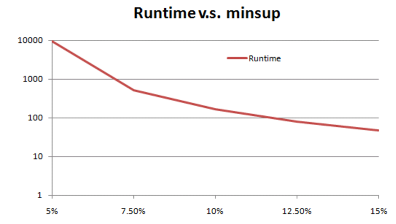
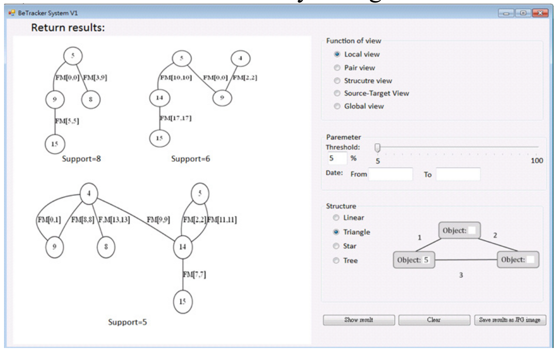


Figure 5. Runtime versus *Minsup* for real dataset

Measuring pattern algorithm's efficiency is very important since we are performing real-time track and analysis for user-given query. In the experiment, therefore, we conduct the experiments to show the execution time of the behavioral mining in our BeTracker. Figure 5 presents the runtime by varying the *minsup* from 5% to 15%. We can find that as the *minsup* gets smaller, the runtime of our methods increases slowly. Even with low *minsup*, our pattern mining algorithm is still efficient. Such great efficiency makes our BeTracker be more useful and applicable for real-time behavior analysis.

VI. System Interface and Plan

We show the interface of our BeTracker system in Figure 6. Our will demo how the temporal subgraph patterns are returned by specifying various user preferences, such as thresholds, relational structure, and the relational structure with partially assigned individuals and sequential time order. Those behavioral patterns with higher support scores will be first returned. We will visualize each pattern that conforms to the given query. In addition, we also perform five functional views with different time periods to show how easily to track and summarize the frequent and representative behaviors for any user-given information.


BeTracker System V1

Return results:

Support=8

Support=6

Support=5

Function of view
- [✓ Local view]
- Pair view
- Strucutre view
- Source-Target View
- Global view

Paremeter
Threshold: 5 %

Date: From To

Structure
- Linear
- [✓ Triangle]
- Star
- Tree

Object: 5

Fig. 6 The interface of BeTracker.

VII. Demonstration Cases

In this section, we demonstrate some behavioral patterns discovered by Our BeTracker system. First, given a query with individual 4 and tree structure in Figure 7(a), we find the pattern shown in Figure 7(b): {(*4*, *FM*, *9*, 0, 1) (*5*, *FM*, *14*, 2, 2) (*14*, *FM*, *15*, 7, 7) (*4*, *FM*, *9*, 8, 8) (*4*, *FM*, *14*, 9, 9) (*5*, *FM*, *14*, 11, 11) (*4*, *FM*, *8*, 13, 13)}. We realize that they form a closed community and follow a temporal order to encounter each other.

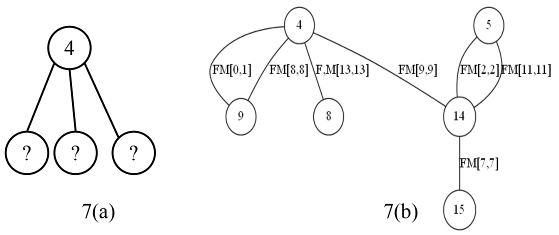


Figure 7. Query tree structure and the result pattern

Given a query with individual 9 and linear structure with sequential time periods shown in Figure 8(a), we find the pattern shown in Figure 8(b): {(*9*, *NFM*, *15*, 0, 4) (*5*, *FM*, *9*, 6, 8) (*5*, *FM*, *8*, 10, 10)}. The object 9 and object 15 are often in the same place for at least four hours but they are not friends in social network. It is because they are co-workers or in the same laboratory, but they are not familiar with each other. After working, object 9 goes to stay with his friend, object 5, for two hours.

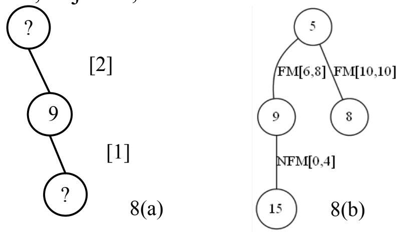


Figure 8. Query linear structure and sequential time periods and corresponding pattern

Moreover, we give a query to in Figure 9(a) to wonder who often likes to stay with object 4 and 5 in the same time. The corresponding pattern, {(*4*, *FM*, *5*, 0, 0) (*4*, *FM*, *15*, 0, 0) (*5*, *FM*, *15*, 0, 0)} shown in Figure 9(b) means they really form a friend group and stay together very often.

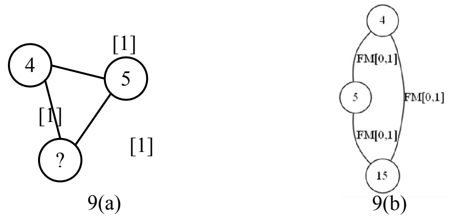


Figure 9. Triangle and simultaneously time periods and corresponding pattern

VIII. Conclusion

Our BeTracker system provides a query-based behavior tracker platform for users to discover human behaviors according to both sensor data and social relationships. The resulting behavioral patterns can not only help understand the complicated human interactions, but also enable effective routing strategy and have better managements of social circles for online social networking services.

Acknowledgements

This work was supported by National Science Council, National Taiwan University and Intel Corporation under Grants NSC99-2911-I-002-001, 99R70600, and 10R70500.

References

[1] S. Albert Ali, P. Eric, T. Romain, G. Theo. T-Patterns Revisited: Mining for Temporal Patterns in Sensor Data. Sensors 10, no. 8: 7496-7513, 2010.

[2] G. Bigwood, D. Rehunathan., M. Bateman, T. Henderson, S. Bhatti. Exploiting Self-Reported Social Networks for Routing in Ubiquitous Computing Environments. IEEE WIMOB 2008, 484-489.

[3] E.Bulut, B.K. Szymanski, Friendship Based Routing in Delay Tolerant Mobile Social Networks. IEEE GlobalCom 2010, 1-5.

[4] P. Hui, J. Crowcroft, E. Yoneki. Bubble Rap: Social-based Forwarding in Delay Tolerant Networks. ACM MobiHoc 2008, 241-250.

[5] N. Pasquier, Y. Bastide, R. Taouil, L. Lakhal, Discovering frequent closed itemsets for association rules, Proceedings of the 7th International Conference on Database Theory, Jerusalem, Israel, 1999, pp. 398-416.

[6] CRAWDAD dataset, http://www.crawdad.org/index.php