SplitterGUI

GUI Application of Mining Fine-Grained Sequential Patterns in Semantic Trajectories

Tanjung Dion, Fawwaz Dzaky Zakiyal
Department of Electrical and Computer Engineering
Pusan National University
Busan, Republic of Korea
{tanjung.dn, dzakybd}@gmail.com

***Abstract*—Semantic trajectory is a sequence of timestamped places wherein each place has information about spatial location and a semantic label. By mining fine-grained pattern that satisfy semantic consistency, spatial compactness and temporal continuity, it will give benefit such as urban planning and targeted advertising. One of the state-the-art method to mine the fine-grained pattern is Splitter algorithm. Firstly, it find set of coarse patterns and their snippets by using PrefixSpan, and then find the set of fine-grained patterns using mean shift method. In this project we build, SplitterGUI, GUI application to visualize the process of the Splitter algorithm. We also discuss the key techniques in Splitter and conduct an experiment using 4SQ database to demontrate the result of SplitterGUI.**

***Index Terms*—fine-grained pattern, semantic trajectories, GUI application.**

I. Introduction

The sequential patterns can be found in a trajectory database, if it properly extracted. It become benefit for targeted advertising, urban planning, location prediction, etc. In other hand, classic sequential pattern mining algorithms cannot work effectively in semantic trajectories, a sequence of time-stamped places wherein each place has information about spatial location and a semantic label. Because of the places in the continuous space cannot be regarded as independent items, so independent place cannot be used to find the frequent sequences. Instead, semantic label need to be grouped to form frequent sequential patterns. But, there is also a challenge to get fine-grained sequential patterns, that it must fulfill requirements of spatial compactness (compact area for each place category), semantic consistency (consistent place ID in each place category) and temporal continuity (limited time constraint).

The process of mining fine-grained pattern have been proposed by Zang et. al [1]. They provide two-step approach to obtain fine-grained sequential patterns. First-step, obtain the coarse patterns from the semantic trajectory data using the full projection method to modify PrefixSpan. The result patterns will fullfil the semantic and temporal constraints. Second-step, splitting each coarse patterns into fine-grained patterns that fulfill the spatial constraint through top-down pattern discovery process using weighted snippet shift by Epanechnikov kernel.

The SplitterGUI project files can be downloaded at $github.com/dzakybd/Datastream-project$

In this project, aiming at implementing the mining fine-grained pattern method proposed by Zang et. al [1] and build the GUI application to visualize the process of the Splitter algorithm, called SplitterGUI. It make the database, course patterns and fine-grained patterns into a visualized map by determined Splitter algoritm's parameters. Afterwards, we discuss the key techniques in Splitter and conduct an experiment using 4SQ database to demontrate the result of SplitterGUI.

II. Implementation

A. System Design

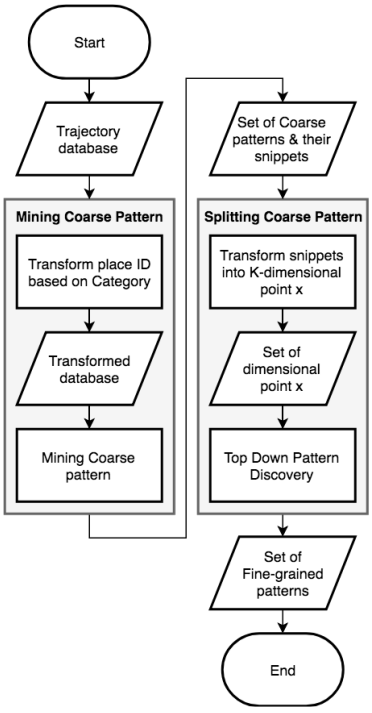

Fig. 1. System Design

Figure 1 show the process of mining fine-grained pattern in semantic trajectories. This method consist of two part: mining coarse pattern and splitting coarse pattern. Mining coarse pattern is a process to find sequential pattern that satisfy semantic consistency and temporal continuity. In the

first step, we add trajectory database that have information about place id, movement time, group id, and spatial location to the system. This trajectory database will be transformed into semantic trajectory database by grouping place id based on category. However, we still keep the original place id in each category to prevent one trajectory from being counted repeatedly. By doing this process, our trajectory database will satisfy semantic consistency.

Then, PrefixSpan algorithm will be used to mining the coarse pattern. In the PrefixSpan algorithm, we include all post-fixes to avoid missing pattern under the given time constrain. Output from mining coarse pattern is set of coarse pattern that will be used in the next process. The next process is finding fine-grained pattern that satisfy spatial compactness in by splitting a coarse pattern in top down manner. Set of coarse pattern from previous process will be transformed into K-dimensional point x.

Then, we employ mean shift clustering to extract the dense and compact snippet cluster based on the support and variance threshold. To speed up the process, the unqualified snippet cluster will be organized into several disjoint communities. By only clustering each communities, we can gradually reduce the kernel windows in clustering method to speed up the clustering process.

B. Key techniques in Splitter

1) Mining Coarse Pattern: Before mining coarse pattern, trajectory database D should be transformed to semantic trajectory by grouping each place based on its category. The purpose is to find any frequent sequences that have similar categories. For example, Fig. 2 shows database transformation with office $= G_1$, shop $= G_2$, restaurant $= G_3$.

Obj.	Semantic Trajectory
O_1	<($G_{1<P3>}$, 0),($G_{1<P1>}$, 10), ($G_{2<P7>}$, 30), ($G_{3<P9>}$, 40)>
O_2	<($G_{1<P2>}$, 0) ,($G_{2<P7>}$, 30), ($G_{1<p5>}$, 130), ($G_{2<P4>}$, 160), ($G_{3<P9>}$, 170)>
O_3	<($G_{1<P3>}$, 0),($G_{2<P4>}$, 30), ($G_{3<P6>}$, 60)>

Fig. 2. Semantic trajectory database

After transformation, we mining sequential pattern that satisfy support threshold σ and time constraint Δt using PrefixSPan algorithm. PrefixSpan algorithm use frequent item as short pattern and prefixes to build projected database then grow the short pattern by searching local frequent item in projected database. However, PrefixSpan algorithm should be modified to satisfy our coarse pattern. 3 shows the modified prefix span algorithm. First, we extract all frequent item at least σ in semantic trajectory D. Then for each item we build item-projected database as S using full projection. Full projection means that we extract all pattern that occurred in database. For example, object 0 in 2 has 2 pattern $P_3 \rightarrow P_7 \rightarrow P_9$ and $P_1 \rightarrow P_7 \rightarrow P_9$.

Output this item as our short pattern and then grow this pattern by calling PrefixSpan function. This function has similar procedure with initial projection function. However, we check time constraint when searching frequent item in projected database. We also call this PrefixSpan function recursively to grow the pattern until the pattern cannot be grown anymore. Output of this algorithm is list of coarse pattern with its snippet (i.e. $P_1 \rightarrow P_7 \rightarrow P_9$), group (i.e. $G_1 \rightarrow G_2 \rightarrow G_3$), and pattern length (i.e. 3).

```
Algorithm 1 : Mining Coarse Pattern
Input : transformed semantic database D, support threshold σ, temporal
constraint Δt
1  Procedure InitialProjection(D, σ, Δt)
2    L ← Frequent items in D at least σ;
3    foreach item i in L do
4      S ← {} ;
5      foreach trajectory o in D
6        R ← postfixes all occurrences of i in o;
7        S ← S U R
8      Output i and its snippets;
9      PrefixSpan(i,1,S|i,Δt);
10 Function PrefixSpan(α, l, S|α, Δt)
11   L ← Frequent item in S|α at least σ and Δt;
12   foreach item i in S|α do
13     α' ← append i to α;
14     S ← {} ;
15     foreach trajectory o in D
16       R ← postfixes all occurrences of i in o;
17       S ← S U R
18     Output α' and its snippets;
20     PrefixSpan(α',1+1,S| α', Δt);
```

Fig. 3. Mining coarse pattern algorithm

2) Finding Fine-Grained Pattern: The result of coarse pattern has satisfy support threshold σ and time constraint Δt. Next step is checking the spatial variance of place in each group category. For example, graph representation of coarse pattern is shown in Fig. 4. Spatial location of place P_5 with other place in group G_1 is to far. Therefore, we have to prune pattern $P_5 \rightarrow P_4 \rightarrow P_6$ from fine-grained pattern. For

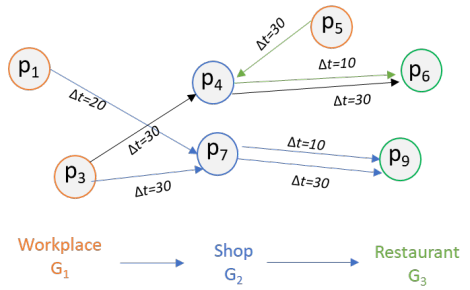

Fig. 4. Graph representation of coarse pattern result

finding fine-grained pattern, mean shift clustering algorithm is used because it does not need any prior knowledge and only use one parameter called bandwidth. Bandwidth parameter is used to limit the area of clustering. The main idea of mean shift algorithm is choosing random point as the center and use bandwidth to limit the coverage area of clustering. Then calculate mean shift vector to shift the center point toward new point that more dense than previous point.

III. Demonstration

A. Database

This project will use the real database, 4SQ that collected from Foursquare check-in sequence in New York. It consists of 48564 places that divided into 417 place categories and it stored semantic trajectories from 14909 users. 4SQ database provides 3 files:

1) *Sequences*, it contains users semantic trajectories including attributes of check-in time-stamp and place ID.
2) *Places*, it contains information about the places (place ID, latitude, longitude, category ID).
3) *Category*, it contains information of place category ID corresponding with its place category name.

B. Result

We will re-implement the process of mining fine-grained pattern using java programing language. The user interface of program will be provided as shown in the Figure. Figure shows the illustration of our program. There are an database input in csv format, 4 parameters input, show button for showing graph representation like in the figure, and 3 button to start the process individually or start all of the process respectively.

Acknowledgment

This paper is submitted to fulfill term project task from Stream Database class (Fall 2018), Pusan National University.

References

[1] C. Zhang, J. Han, L. Shou, J. Lu, and T. La Porta, "Splitter: Mining fine-grained sequential patterns in semantic trajectories," *Proceedings of the VLDB Endowment*, vol. 7, no. 9, pp. 769–780, 2014.