J Ambient Intell Human Comput (2018) 9:1345–1357
DOI 10.1007/s12652-017-0519-z

ORIGINAL RESEARCH

Prediction of next destinations from irregular patterns

Mehdi Boukhechba[1] · Abdenour Bouzouane[1] · Sébastien Gaboury[1] · Charles Gouin-Vallerand[2] · Sylvain Giroux[3] · Bruno Bouchard[1]

Received: 23 September 2016 / Accepted: 31 May 2017 / Published online: 29 June 2017

Abstract Few decades ago, understanding human behaviors was considered as a mystery where predicting people's future was impossible. Many changes have been noticed since that era. Thanks to current advances in location tracking technology and data mining techniques, predicting users' behaviors has become possible. In this paper we present a new algorithm to online predict users' next visited locations that not only learns incrementally the users' habits, but also detects and supports the drifts in their patterns. Our original contribution includes a new algorithm for online association rules mining that supports the concept drift.

Keywords Human activities · Activity prediction · Online association rules · Concept drift

✉ Mehdi Boukhechba
Mahdi.Boukhechba1@uqac.ca

Abdenour Bouzouane
Abdenour_Bouzouane@uqac.ca

Sébastien Gaboury
Sebastien_Gaboury@uqac.ca

Charles Gouin-Vallerand
charles.gouin-vallerand@teluq.ca

Sylvain Giroux
sylvain.giroux@usherbrooke.ca

Bruno Bouchard
Bruno.Bouchard.PhD@ieee.org

[1] LIARA Laboratory University of Quebec at Chicoutimi (UQAC), Chicoutimi, QC, Canada

[2] Tele-universite of Quebec (TELUQ), Montreal, QC, Canada

[3] University of Sherbrooke, Sherbrooke, QC, Canada

1 Introduction

The last decade has been the mobile device technologies era where the capture of the evolving position of moving objects has become ubiquitous. Mobile wearable tracking devices, e.g., phones and navigation systems collect the movements of all kinds of moving objects, generating huge volumes of mobility data. Despite the fact that data collected from mobile devices is more accurate, there are still scientific locks regarding the use of this data. One of these important research topics is the prediction of users' next locations (Gambs et al. 2012; Kapterev 2014; Li and Fu 2014; Zheng et al. 2008) that proposes a set of services used in a wide range of fields, such as traffic management, public transportation, assistance of people with special needs, commercials, and advertising.

Our precedent work (Boukhechba et al. 2016) proposed an online activity recognition system that offers the possibility to understand what people are doing at a specific moment by inferring incrementally their visited places from raw GPS data. In this paper, we are proposing an evolution of our precedent system that estimates the users' actions in the future by predicting their next visited point of interest. We are planning to use such a system to assist people with special needs (e.g. persons suffering from Alzheimer disease) during their daily outdoor activities by proposing specific assistance based on their recognized activities and context. The predictive model that we are proposing in this paper aims to launch assistance processes when users are lost by suggesting a new safe destination.

In this paper, we are addressing the issue of predicting the next location of an individual based on the observations of his mobility habits. One of the major problems met when trying to incrementally learn users' routines is the concept drift (Gama et al. 2004, 2009; PhridviRaj and

GuruRao 2014). The concept drift means that the statistical properties of the target variable (in our case, the users' habits), which the model is trying to predict, change over time in unexpected ways. For instance, assuming that we learn a user's habits using a traditional algorithm, a user lives in "address 1" for 1 year, after that he moves to "address 2", this shifting will lead to not only shift the address but probably the habits too. Existing algorithms will take few months to detect that the user is doing new habits (if we take a support of 60% using Apriori[1] algorithm (Gai 2012) for example, it will take 7 months to detect the new habits). In the meantime, all the next locations proposed by these algorithms are probably false because the predictions are based on the old routines and not the new ones.

Current works Boukhechba et al. (2015), Gama et al. (2009), Gambs et al. (2012) and Hipp et al. (2000) that try to learn and predict users' routines fail in the ability to deal with the changes in users' behaviors. Moreover, there are only few works (Kapterev 2014; Simmons et al. 2006) that attempt to incrementally predict the users' next location.

We bring a novelty to the manner of resolving this problem via a novel online algorithm that extracts association rules carrying the data drift during the learning process. Hence, the main idea is to help the new detected habits to become quickly frequent. To do so, we introduce a new criterion of support calculation based on a weight distribution of data collected rather than the classic number of occurrences.

The following sections detail our contribution: Sect. 2 reviews related work; Sect. 3 presents our approach; Sect. 4 describes the experimentation. Finally, conclusion and future works, as well as the expected contributions, are summarized in Sect. 5.

2 Related works

Significant research effort has been undertaken in both mobile computing and spatial data mining domains (Spaccapietra et al. 2008; Zheng et al. 2008). Many advances in tracking users' movements have emerged resulting in several proposals for predicting future users' locations. The main approach proposed is to learn a user's patterns from his historical locations and try to predict the next location via different techniques.

In Morzy (2006), Morzy introduces a new method for predicting the location of a moving object where he extracts the association rules from the moving object database using a modified version of Apriori, he uses the rules extracted when a trajectory is given via matching functions, he selects the best association rule that matches this trajectory, and then uses it for the prediction. Unfortunately, this work does not permit incremental models' training, since it is based on a posteriori learning. Secondly, the fact that authors propose matching functions in the form of strategies (simple, polynomial, logarithmic and aggregation strategies) can create computational complexities; difficulties to choose and set the right parameters for the right strategy.

Sébastien et al. extended a previously proposed mobility model called the Mobility Markov Chain (n-MMC),[2] in order to keep track of the n previous locations visited (Gambs et al. 2012). This proposal essentially corresponds to a higher order Markov model. Authors show that while the accuracy of the prediction grows with n, choosing $n > 2$ does not seem to bring an important improvement to the cost of a significant overhead in terms of computation and space for the learning and storing of the mobility model. However, like the previous works, this one has a lack with the computational complexity and the incremental support. In addition, the three datasets used in authors' experiments were collected in a controlled environment where data was gathered from specific participants who were aware of the experiments.

Asahara et al. (2011) proposed a method for predicting pedestrian movement on the basis of a mixed Markov-chain model (MMM),[3] taking into account some complex parameters like pedestrian's personality merged to his previous status. The authors experiment their solution in a major shopping mall and report the accuracy of 74.4% for the MMM method and, in a comparison over the same dataset, they reported that methods based on Markov-chain models, or based on Hidden Markov Models, achieve lower prediction rates of about 45 and 2%, respectively.

Authors in Ezeife and Su (2002) present two new algorithms that use the frequent patterns tree (FP-tree)

[1] Apriori is an algorithm for frequent item set mining and association rule learning over transactional databases. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database.

[2] MMC is a probabilistic automaton in which states represent points of interest (POIs) of an individual and transitions between states corresponds to a movement from one POI to another one, a transition between POIs is non deterministic but rather that there is a probability distribution over the transitions that corresponds to the probability of moving from one POI to another.

[3] MMM is an intermediate model between individual and generic models. The prediction of the next location is based on a Markov model belonging to a group of individuals with similar mobility behavior. This approach clusters individuals into groups based on their mobility traces and then generates a specific Markov model for each group. The prediction of the next location works by first identifying the group a particular individual belongs to and then inferring the next location based on this group model.

structure to reduce the required number of database scans. One of the proposed algorithms is the DB-tree algorithm, which stores all the database information in an FP-tree structure and requires no re-scan of the original database for all update cases, the algorithm stores in descending order of support all items and counts all items in all transactions in the database in its branches. The DB-Tree is constructed in the same way as done in FP-Tree except that it includes all the items instead of only the frequent 1-items. The second algorithm is the PotFp-tree (Potential frequent pattern) algorithm, which uses a prediction of future possible frequent item sets to reduce the number of times the original database needs to be scanned when previous small item sets become large after database update. The first disadvantage of the three algorithms, with all respect to the authors, is the non-support of the concept drift, since none of them support the changes in the sequences behavior. The second problem is the restructuring of the tree to store the node in descending order of support, this technique not only increases the computational complexity, but represents likewise an invalid solution to fields where the order of items is important like peoples' habits.

In Katsaros et al. (2003), a method called dynamic clustering based prediction (DCP) of mobile user movements is presented to discover user mobility patterns from collections of recorded mobile trajectories and use them for the prediction of movements and dynamic allocation of resources. Collected user trajectories are clustered according to their in-between similarity using a weighted edit distance measure (Hipp et al. 2000). In the prediction phase, the representatives of the clusters are used. Authors showed using a simulation that for a variety of trajectory length, noise and outliers, the DCP method achieves a very good tradeoff between prediction recall and precision.

While developing a rich body of work for mining moving object data, the research community has shown very little interest for the online mining of these objects since the mainstream of related works lies on a post-hoc analysis of a massive set of data to learn and predict locations.

Moreover, one of the big issues that can easily shatter the most robust next location predictive model is the habits' drift, since from the time when the data begins to behave in a non-regular manner; the predictive models will face difficulties to do their work. Finally, to our knowledge, we are the first to support incrementally the users' habit changes, since there is no approach that handles the habits' drift during the learning and the prediction of next locations process.

3 Overview of the approach

Our solution learns users' habits by analyzing their visited places, these visited places are called POIs. A Place of Interest (POI) by definition is an urban geo-referenced object where a person may carry out a specific activity. Our approach begins by constructing a sequence of POI_i that represents the tracking of users' daily habits, every sequence is stored incrementally in a tree structure called Habits' Tree 'HT'. On every sequence arrival, our algorithm checks for a drift in the distribution of sequences and allocates a new weight to the sequence concerned, when achieved, the new sequence is added to the user's HT and finally, the algorithm predicts the next POI using the association rules drawn from HT (see Fig. 1).

In the following sections, we are going to present every part of our model starting by the sequence construction step.

3.1 Sequence construction

Users' habits are composed of daily routines that define the pattern of users' movements in the city. This step aims to represent users' habits via a series of routines called sequence S_i, every sequence contains a set of POI_i. POI extraction is a pretreatment task used to switch people's raw GPS data to meaningful semantic labels (e.g. Home, Restaurant, University, etc.). Note that this step may include errors due to the noisy raw GPS data. In this work we use a spatiotemporal clustering algorithm (Boukhechba et al. 2016) that reported an accuracy around 86%.

S_i contains a set of disjoint singletons POI_i (i.e. we cannot find the same POI many times in the same sequence) and terminates at the end of the day (daily habits). For example, assuming that the user achieved the following activities during a day: Home, Work, Restaurant, Work, Gym, Home; the algorithm will construct incrementally two sequences from these habits:

S_1: Home, Work, Restaurant.

S_2: Restaurant, Work, Gym, Home.

In fact, when Algorithm 1 detects a new POI that already exists in the sequence, like "work" in the example above, it stops constructing S_1 and creates a new sequence S_2, the reason of our proceeding is to optimize the storage of the sequences (this point will be discussed further). The st POI of S_1 will be the first POI of S_2 in order to make a link between the two sequences, if this is link is not made, we would not know the user's next destination after restaurant.

The sequences S_i are stored in a Habits' tree HT, it is a new data structure that we proposed and that takes form of a special tree. In HT, every node represents a POI and is characterized by a weight w_i that represents

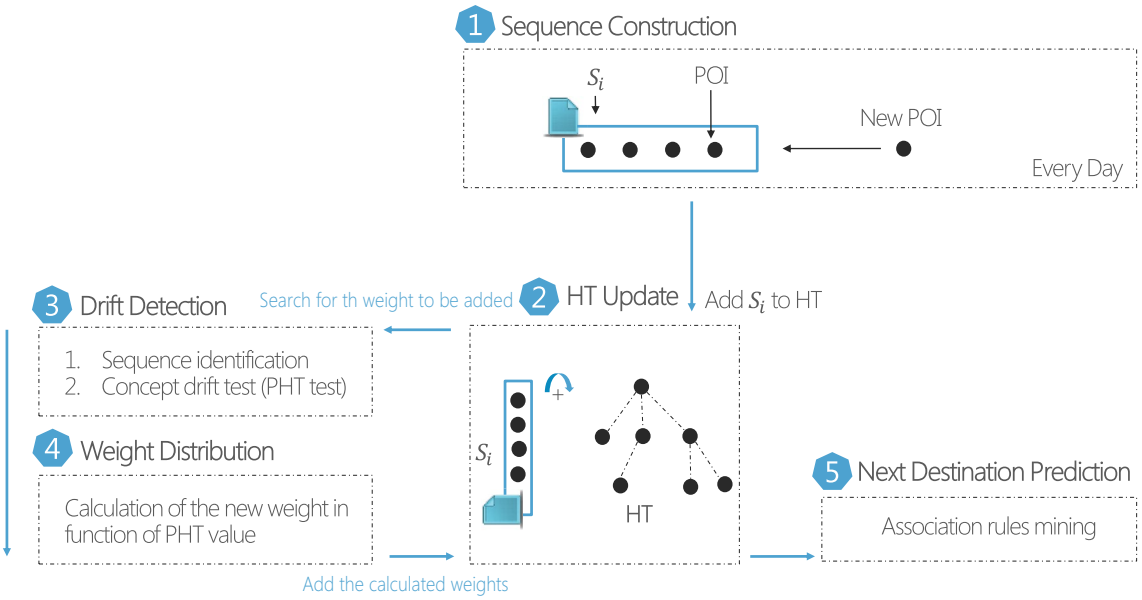

Fig. 1 The overall approach of our predictive model

the weight of POI_i's in HT, moreover, every node has an identifier ID_i that aims to identify the sequences by an integer (see Sequence identification in Sect. 3.2).

To give a brief idea of the way the data are structured in HT, we illustrated in Fig. 2 how our algorithm stored the two sequences S_1 and S_2 from the example above in a form of a succession of nodes characterized by <name,w, ID>.

Algorithm 1: Sequence construction

Input: A POI_j;
Output: Sequence S_i;

S_i = null ;
1: **For each** new POI_i
2: **If** (! S_i.contains(POI_i) and StillTheSameDay) then $S_i = S_i + POI_i$
3: **Else** //new sequence
4: **Return** S_i;
5: **If** (StillTheSameDay) S_i = Last POI_i
6: **Else** S_i = null ;
7: **End for each**

Algorithm 1 is executed on every arrival, when is finally constructed we move to the next step: the habits' tree update.

Note that an improvement can be made at this step to respect the definition of streaming learning, which means that every new POI is treated instantly without waiting for the construction of S_i, this improvement will be proposed in our future works.

3.2 Habits' tree update

The work of Ezeife and Su (2002) inspired us to plan this part, authors in that work proposed a new algorithm for mining incrementally association rules called DB-Tree. DB-Tree is a generalized form of FP-Tree [FP-Growth

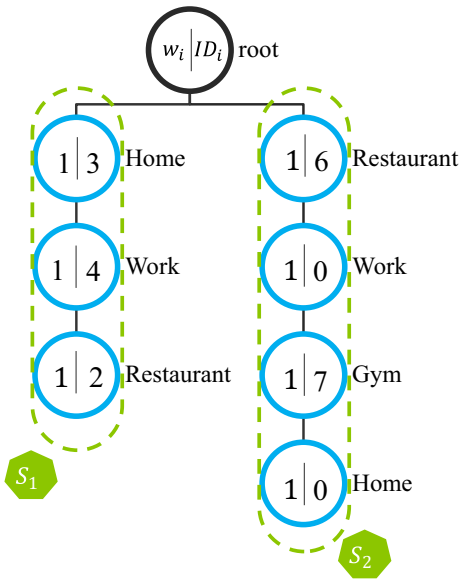

Fig. 2 HT structure construction. Every node represents a POI and contains a weight w_i on the *left* and an identifier ID_i on the *right*. The ID_i of each node is a value between 0 and 9. Note that ID_i is not unique because we aim to identify the sequences and not the POIs. The sequence identification process is discussed in Sect. 3.3

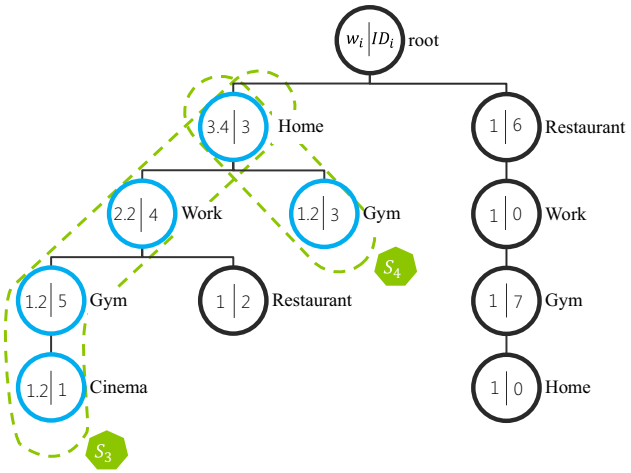

Fig. 3 Sharing paths in HT structure

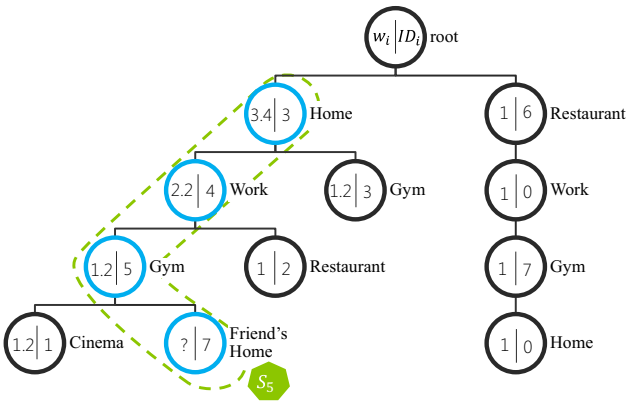

Fig. 4 An example of HT updated

(Hipp et al. 2000)] which stores in descending order of support all items in the database and counts all items in all transactions in the database in its branches. The DB-Tree is constructed in the same way as done in FP-Tree except that it includes all the items instead of only the frequent 1-items.

In our habits' tree, tree data structure was not chosen arbitrarily. By using this structure and by storing all the sequences (frequents and not frequents) we eliminate the need of rescanning the entire database to update the structure like it is done in Apriori and FP-tree (when previously not frequent POI_j become frequent in the new update). Indeed, the algorithm scans only the branches concerned by the new sequence, which optimizes the computational complexity. Additionally, storage optimizations are achieved using this structure, sharing paths between items in tree structure leads to much smaller size than that in a traditional database (see Figs. 2, 3, 4).

Let us take the following example: we take the HT presented in Fig. 2 and we add two new sequences:

S_3 = Home, Work, Gym, Cinema.

S_4 = Home, Gym.

From the updated HT presented in Fig. 3, we can observe how the notion of sharing paths (in the nodes home and work) leads to a compression of the database dimension (this characteristic will be discussed deeper in the experimental section). Additionally, we can notice that the algorithm added these sequences with different weight than the previous because it detects new behaviors, more details will be provided in the next sections.

On the arrival of a new sequence S_i, the Algorithm 2 recursively processes each POI_i in S_i. If the POI_iexists in HT, the concerned node's weight is updated, otherwise, the algorithm adds a new node with a new random ID_i, and a new weight w_i where the details of calculation is given in the next section.

For example, supposing that, after a certain time of learning, a user's HT is structured like in Fig. 3, the next day, the user did the following sequence:

S_5= Home, Work, Gym, Friend's Home.

Figure 4 shows how our algorithm updates the nodes: Home, Work, Gym; and adds a new POI: Friend's Home. Note that Friend's Home was added to HT with an unknown weight because it will be calculated in the next section.

So after, seeing how we proceed to structure HT on every S_i arrival, it is time to present how we calculate the weight that will be added in every node. Actually, the weight is calculated in two ways depending on if the new S_i is frequent or not, the formula is given as follows:

$$\begin{cases} w_i = 1 + wd_i, & \textit{if } S_i \textit{ is not frequent} \\ w_i = 1, & \textit{if } S_i \textit{ is frequent} \end{cases}$$

The drift's weight wd_i is an extra weight that will be added if the S_i is considered as a not frequent habit. As we see, our distribution behaves in two ways: a traditional way when the sequence S_i is frequent (adding only 1 in every node), and a special way when S_i is not frequent (adding $1 + wd_i$), see rows [4, 9] in Algorithm 2. The reason why we proceed this way is that we are trying to help only the new habits that are not frequents to become frequents, once arrived, we stop our help not to promote a sequence (habit) relative to another.

Algorithm 2: Habits' tree update

Input: A sequence S_i;
Output: HT;

```
For each (POI in S_i )
    HT.add POI
End for each
If (S_i is frequent )    w_i = 1;
Else                     w_i = 1+ Extra weight distribution (S_i);
HT.add Weights (w_i);
```

The next section presents how we detect the drift in the user habits in order to calculate this extra weight wd_i. Our contribution aims to track the drift in users' habits, and to distribute the weights based on these behavior changes.

3.3 Drift detection

This part aims to track changes in users' habits, or in other terms, it aims to check if the new sequence is an old or a new routine. Our technique is divided into two steps: firstly we formulate mathematically each new sequence and secondly we use this number to test if this sequence is new or not (it is a concept drift).

Sequence identification In order to be able to use a concept drift test, it is crucial to parse the information contained in S_i into a quantifiable entity, which means, instead of comparing a sequence of strings, we are going to parse every sequence into a number to facilitate comparison.

In fact, the introduction of ID_i in each node was in this perspective. Every new sequence will be represented by a variable called *xi* where *xi* is obtained by the concatenation of each ID_i present in S_i.

For instance, in the example cited in Fig. 2, if we want to calculate the mathematical representation of $S_2 =$ *Restaurant*, *Work*, *Gym*, *Home*, the variable $x2$ will take a value of 6070 (i.e. $ID = 6$ for Restaurant, $ID = 0$ for Work, $ID = 7$ for Gym, $ID = 0$ for Home). Similarly, the identifier of S_1 is $x1 = 342$ (i.e. $ID = 3$ for Home, $ID = 4$ for Work, $ID = 2$ for Restaurant).

Once the sequence is identified by an integer, we use this number in the next step, the concept drift test.

Concept drift test We use the Page Hinkley Test (PHT) (Hipp et al. 2000) to detect changes in users' habits, PHT is a sequential analysis technique typically used for monitoring change detection. It allows efficient detection of changes in the normal behavior of a process which is established by a model. The PHT was designed to detect a change in the average of a Gaussian signal (Gama et al. 2009). This test considers a cumulative variable U_T defined as the cumulated difference between the observed values (in our case the sequences' identifier *xi*) and their mean till the current moment.

The procedure consists of carrying out two tests in parallel. The first makes it possible to detect an increase in the average. We calculate then:

$$\begin{cases} U_t = \sum_{d=1}^{t} \left(x_d - \bar{x}_d - \delta\right),\ U_0 = 0 \\ m_t = \min\left(U_t\right),\ \ t \geqslant 1 \\ PHT = U_t - m_t \end{cases} \tag{1}$$

The second allows detecting a decrease in the average as follows:

$$\begin{cases} U_t = \sum_{d=1}^{t} \left(x_d - \bar{x}_d + \delta\right),\ U_0 = 0 \\ M_t = \max\left(U_t\right),\ \ t \geqslant 1 \\ PHT = M_t - U_t \end{cases} \tag{2}$$

where $\bar{x}_d = \left(\sum_{d=1}^{t} x_d\right) \Big/ t$ and δ corresponds to the magnitude of changes that are allowed. When the difference PHT is greater than a given threshold (λ) a change in the distribution is assigned. The threshold λ depends on the admissible false alarm rate. Increasing λ will entail fewer false alarms, but might miss or delay some changes. Controlling this detection threshold parameter makes it possible to establish a trade-off between the false alarms and the miss detections. In order to avoid issues linked to the parameterization of λ, we were inspired by the work of Zhang et al. (2010), where authors propose a self-adaptive method of change detection by proving that λ_t can be self-adapted using this equation: $\lambda_t = f \times \bar{x}_t$ where f is a constant called the λ factor, which is the number of required witnesses seeing the changes.

In our case, habits' drift that we are tracking are brutal, in the sense that the user does not usually change his habits gradually, so, we do not need a large number of witnesses to detect the changes, that's why we put $f = 2$ in case of increasing average and $f = 1/2$ in case of decreasing average.

For instance, assuming that the user has been doing the sequence S_5 = Home, Work, Gym, Friend's Home for 10 days, after that he changes his habit from S_5 to S_4 = Home, Gym (see Figs. 4, 5).

We are going to illustrate the detailed calculation of PHT after the change of habit (the 11th day). First, we need to represent the two sequences mathematically, as the concatenation process seen before, $x5_{10} = 3457$ and $x4_{11} = 33$ represent the representation of S_5 in the 10th day and S_4 in the 11th day respectively (see Figs. 4, 5). The new habit number $x4$ is less than $x5$, so we are tracking a decrease of the average, consequently we use the Eq. (2) as follows :

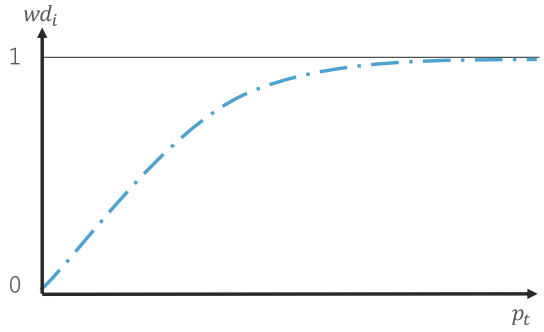

Fig. 5 Mathematical representation of wd_i

$$\begin{cases} U_{11} = \sum_{d=1}^{11} (x_d - \bar{x}_d + \delta),\ U_0 = 0 \\ M_{11} = \max(U_d),\ t \geqslant 1 \\ PHT = M_{11} - U_{11} \end{cases} \begin{cases} U_{11} = U_{10} + \left(x4_{11} - \overline{x4_{11}} + \delta\right) \\ M_{11} = \max(U_d),\ t \geqslant 1 \\ PHT = M_{11} - U_{11} \end{cases}$$

As during the 10 days before the change of habit, the user was doing the same habits:

$$U_{10} = \sum_{j=1}^{10} \left(x5_j - \overline{x5_j} + \delta\right) = \sum_{j=1}^{10} (3457 - 3457 + \delta) = 10\delta$$

As said before δ represents the minimum of changes allowed, in our case $\delta = 1$, so $U_{10} = 10$.

$$\begin{cases} U_{11} = 10 + (33 - 3457 + 1) \\ M_{11} = \max(U_d),\ t \geqslant 1 \\ PHT = M_{11} - U_{11} \end{cases} \begin{cases} U_{11} = -3413 \\ M_{11} = 10 \\ PHT = 3423 \end{cases}$$

After calculating the threshold $\lambda_{11} = 1/2 \times \overline{x5_{10}} = 1728.5$, we notice that we are in the presence of a change of habits because $PHT > \lambda_{11}$.

After having a value (PHT) that represents the stability of our users' habits, we are going to use this variable to distribute the weight in every new POI's node in HT.

3.4 Extra weight distribution

As said earlier, our approach behaves in two ways (see Algorithm 2): a traditional way when the sequence S_i is frequent (adding only 1 in every node), and a special way when S_i is not frequent (adding $1 + wd_i$), the extra weight wd_i is calculated using an exponential function as follows:

$$wd_i = 1 - e^{-\frac{1}{2}\frac{PHT}{\lambda_t}}$$

where $p_t = \frac{PHT}{\lambda_t}$ represents an indicator of the user's state, the greater is PHT than λ_t more we are sure that the user is doing something new, and vice versa. For example, when there are no new habits in the user behaviors $PHT = 0$, so $p_t = wd_i = 0$, see Fig. 5. By against, each time p_t approaches the value 1, we conclude that the user has a drift in his habits.

For instance, in the past section we calculated the $PHT = 3423$ value on the 11th day and we found that it exceeds the threshold $\boldsymbol{\lambda}_{11} = 1728.5$, in this case the weight added will be calculated as follows:

$$\begin{cases} w_{11} = 1 + wd_{11} \\ wd_{11} = 1 - e^{-\frac{1}{2}\frac{3423}{1728.5}} = 0.62 \end{cases}$$

Algorithm 3: Extra weight distribution

Input: sequence S_i;
Output: Weight w_i ;

1: xi = HT.GetSequenceId(S_i);
2: Calculate λ_t, U_T, m_t, M_t;
3: Calculate PHT;
4: Calculate wd_i;
5: **Return** wd_i

So the weight that will be added to HT will be $w_{11} = 1.62$.

After calculating the weight that will be added on S_i arrival, the next step is to get the association rules from HT to predict the next activity.

3.5 Next destination prediction

Association rules mining Our technique is inspired by FP-growth algorithm and one of its incremental versions called DB-Tree (Ezeife and Su 2002). The main difference between our work and theirs is that we introduced a weight distribution function that tracks the habits drift. Secondly, our structure does not order the tree's items in descending order of support like done in DB-tree. In fact, DB-tree and FP-growth do not make a difference between "home, work, restaurant" and "restaurant, work, home". Clearly, we cannot use such technique when analyzing users' habits, otherwise, it will lead to gross errors.

Suppose we have a database with a set of items like illustrated in Fig. 3, I = {Home, Gym, Work, Restaurant, Cinema} and MinSupport = 60% of database transactions [MinSupport is a support threshold used to identify the frequent data subsets (Gai 2012)]. To compute the frequent POI_i after constructing the habit tree HT, the algorithm mines the frequent POI_i that satisfy the minimum support represented by the MinSupport percentage of the maximum item's weight in HT. From Fig. 3, we have the weight of every item as follows (note that for every item, we add up the corresponding weights contained in the entire tree, for example Home's weight = 3.4 + 1.0 because it appears twice in HT):

I = {(Home: 4.4), (gym: 3.4), (work: 3.2), (restaurant: 2), (Cinema: 1.2)}.

The minimum support will be:$MinSup = \frac{60}{100}$ (4.4) = 2.64, thus, the frequent POI_i are all items greater or equal to 2.64 as {(Home: 4.4), (gym: 3.4), (work: 3.2)}.

The next step is mining the frequent patterns from HT and association rules that are quite similar to those in FP-growth (Ezeife and Su 2002), thus we see no need to repeat the same process explanation.

Next activity After mining the association rules from HT, we get from the same example in Fig. 3 these rules:

$$\begin{aligned} Work,\ Gym &\rightarrow Home \\ Work &\rightarrow Gym \\ Home &\rightarrow Gym \\ Home &\rightarrow Work \\ Home,\ Work &\rightarrow Gym \end{aligned} \tag{3}$$

Predicting the next activity lies in choosing the most appropriate association rules with the highest weight that represent the user situation, and using the resulting clause as predicted next activity. For example, if we know that the user is gone from home to work, using the last association rules (3) we can predict that he will go next to the gym.

Algorithm 4: final algorithm

Input: POI_i; Users past activities UPA;
Output: Next activity;

S_i = Sequence construction (POI_j);
w_i = Extra weight distribution (S_i) ;
HT = Habits 'tree update(S_i);
Return Next location = Activity prediction (HT,UPA);

Finally, we present in Algorithm 4 the whole process that predicts the next activity from users' current location and some of their past locations if exist. First we construct a sequence of POI from the current location (or we wait until the sequence is constructed), then we calculate the weight that will be added to HT and we update the tree using this weight, we search for the association rules and we predict the next location based on the user's past activities (if exist).

Note that the sequence of these steps is not essentially like mentioned in Algorithm 4, we present in this algorithm the whole process to explain how to start from a simple localization to predict the next user's activity. In real life, these processes can be used differently, for example, there is no need to search for the association rules on every sequence arrival, the most correct way is to update HT on every S_i (because the update does need a whole scan of the tree, so it is not expensive in term of calculation), and to search for the association rules in an appropriate time depending on the application requirements. For example, supposing that we try to assist a patient of Alzheimer's disease, the user tends to forget his next activities, the appropriate time that we are talking about is when the proposed system detects an anomaly in the user's behaviors (the user makes mistakes because he does not know what to do next), at this moment the system searches for the association rules in order predict the next probable activity.

4 Experimental evaluation

In the experimentations, we address the following questions: (1) how does our algorithm compare with other states of the art? (2) How does the disparity of habits affect the algorithm results? (3) How does our algorithm behave in a mobile environment?

4.1 Datasets

We evaluate our approach using two types of data: synthetic data and real data.

Synthetic data We asked three users with different profiles to note their daily habits for 3 months, the choosing of users was not arbitrary, we chose them with different habits disparity level : "user 1" with very recurrent habits, "user 2" with moderately recurrent habits and user 3 with very low recurrence level. The dataset contains 107 different activities (POIs) and around 380 sequences.

Real data To push even further the level of our experiment, we used a renowned dataset from the Microsoft research project GeoLife (Zheng et al. 2008). The GPS trajectory dataset was collected in (Microsoft Research Asia) Geolife project by 182 users in a period of over 3 years (from April 2007 to August 2012). A GPS trajectory of this dataset is represented by a sequence of time-stamped points, each of which contains the information of latitude, longitude and altitude. This dataset contains 17,621 trajectories with a total distance of about 1.2 million kilometers and a total duration of 48,000 + h. These trajectories were recorded by different GPS loggers and GPS-phones, and have a variety of sampling rates. 91 percent of the trajectories are logged in a dense representation, e.g. every 1–5 s or every 5–10 m per point. This dataset recorded a broad range of users' outdoor movements, including not only life routines like go home and go to work but also some entertainment and sports activities, such as shopping, sightseeing, dining,

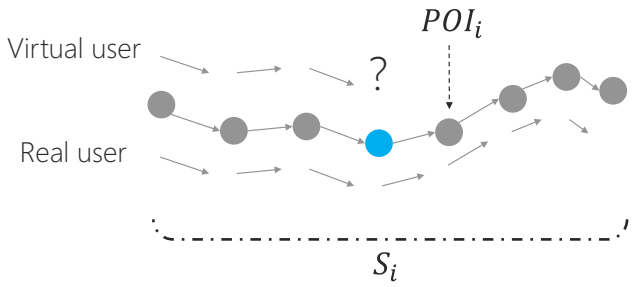

Fig. 6 Testing process with real and virtual users

Table 1 Standard incremental activity prediction results using four different mobility patterns

	Simulated data (107 POIs)			Geolife (2831 POIs)	
	User 1 (%)	User 2 (%)	User 3 (%)	User 4 (%)	User 4 + time
Precision	83	71	61	68	70
GE	17	29	39	32	30
LE	11	14	13	9	7
HE	6	15	26	23	23

The time dimension has been added to User 4 and analysis in the discussion at the end of this section

hiking, and cycling. The final dataset includes 2831 different visited places (POIs).

4.2 Testing process

Test process in association rules represent a delicate step, since how to test is related to the field of study. We created a specific testing process to represent as much as possible the activity prediction situation. The concept is based on the introduction of a second virtual user that will follow the real users' movements all the time, however, during every sequence of POI, the virtual user will have a memory lapse (he will forget his next destination), in this case our algorithm will predict a next localization that will be compared to the real next activity (see Fig. 6).

We assume the memory lapse is random, in fact, on every S_i arrival, our algorithm generates a random position between 1 and S_i's length, this position represents the POI's position where the virtual user will forget his next destination.

The precision represents the number of sequences where activities were well predicted on the total number of sequences, by against, the global error GE, which is calculated using the number of sequences where the activities' predictions were mistaken on the total number of sequences.

The global error contains two types of error: learning error LE and habit error HE. LE represents an error in the prediction of the next activity when referring to the past activities. For example, the user has the habit of going sometimes from home to work and other times to drive his child to school, if we do a test starting from the POI "Home", our algorithm will predict for example work as next destination because it is the most recurrent activity after home. All the time when the user will go driving his child to school and when we predict work as next activity, the algorithm will record a LE (this mismatching problem will be handled in our next work, please see future work section).

HE represents the disability to predict a next activity because the user's precedent POI are not frequent, this error can be seen as a similarity index, the greater is HE, the more data is scattered (there is less recurrence in the user's habits).

We evaluated our approach by highlighting three dimensions: first we tested our algorithm with a standard dataset, secondly with a dataset that contained a concept drift, and finally we tested the performance of our work on the mobile environment.

4.3 Standard incremental activity prediction experiment

In this step we used four users' data, the three from the synthetic dataset and one user from Geolife dataset. Results presented in Table 1 represent the precision, GE, LE and HE of our algorithm on every user data.

From Table 1, our algorithm predicts the next activities of user 1 (with very recurrent habits) with a precision of 83% and a global error GE of 17% divided into 11% of learning error LE and 6% of habit error. User 2 and user 3 show less precision rate with respectively 71 and 61% of precision. User 4 data that contains 726 sequences and a total of 2831 POI (class) shows a precision of 68% and a global error of 32%.

Our approach shows an interesting result with an average precision of 70.75%. Moreover, while analyzing the distribution of errors in every user's data, we notice a strong correlation between the global error GE and the habit error HE, indeed, the variation of learning error LE is so small that we preclude the possibility of linking between GE and LE.

We conclude that the error in our approach is sensitive to the users' habits similarity, which is somewhat logical because the definition word "Habit" is a routine of behavior that is repeated regularly. Correspondingly, the definition of "habits' change" is the alteration in a user's regular routines and not the alteration in the rare ones. Regularity in users'

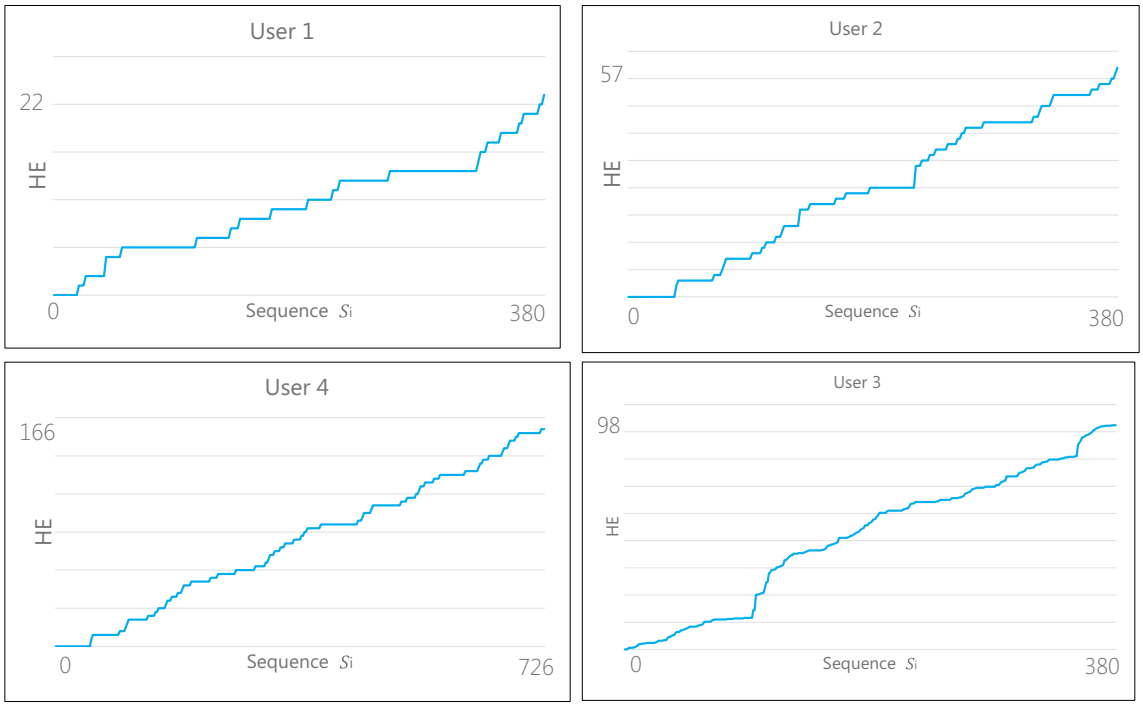

Fig. 7 Habit error HE evolution in the four users' data

movements is important to succeed in predicting their next locations. The more regular are a user's routine, the better is the precision. As such, daily routines are easier to predict than weekly ones, similarly, weekly routines are easier to predict than monthly ones…etc.

In Fig. 7, we track the evolution of HE in every user's data. Results presented show globally two kinds of graphics: stepped graphic concerning user 1 and user 2 owning to the fact that those two users have some new behaviors, when arrived, the algorithm makes a mistake in that moment because the new sequence is unknown, but it catches up quickly in the next moments to recognize the sequence as frequent and predicts the right activity. The second type of graphic concerning user 3 and user 4 is a moderately smoothed graphic which approaches a straight function, the user 3 and user 4 have dispersed habits which explains the continuous increase of HE over time.

Discussion Note that the predictive model can be improved by introducing temporalities in HT. We believe that the time dimension can help reduce GE and LE. However, the temporal dimension has been discarded in this paper to simplify our proposal and focus it on the concept drift supporting technique. To show how temporalities can improve our predictive model, we conducted a small experiment while differentiating the sequences that happen in working days from those in weekends (e.g. the same restaurant is named differently in week days than in weekends). So, we have added the time dimension and analyzed the predictive model results for user 4. Results presented in the last column of Table 1 show how adding this time differentiation can help reduce the model error, specifically, LE that is related to the learning error.

4.4 Incremental activity prediction with concept drift experiment

In this section, we compare our approach with an incremental version of FP-growth called DB-Tree (Ezeife and Su 2002) and Concept-Adapting Very Fast Decision Tree (CVFDT) (Witten et al. 2017), an extension of VFDT algorithm that handles the concept drift.

The ideal dataset to experiment the three algorithms is a dataset where the user has made a relocation (change of address and probably of habits) using a dataset that contains a concept drift, for that, we paired two users' dataset from Geolife users' datasets into one dataset to say that the first user changes his address and his habits to the second user's address and habits, the new dataset contained 389 POIs.

Table 2 presents the results of our experiment; we divided it into three indicators, precision before and after the relocation, and the global precision.

Table 2 Comparison between our algorithm and DB-tree algorithm

	Our algorithm (%)	DB-tree (%)	CVFDT (%)
Precision before the relocation	72	63	60
Precision after the relocation	69	51	60.5
Global precision	70.5	57	60.25

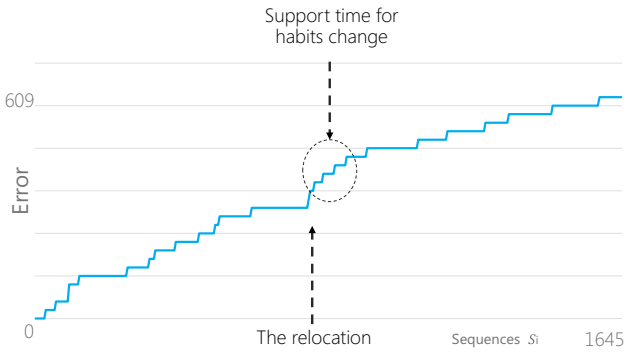

Fig. 8 Error evolution in our algorithm

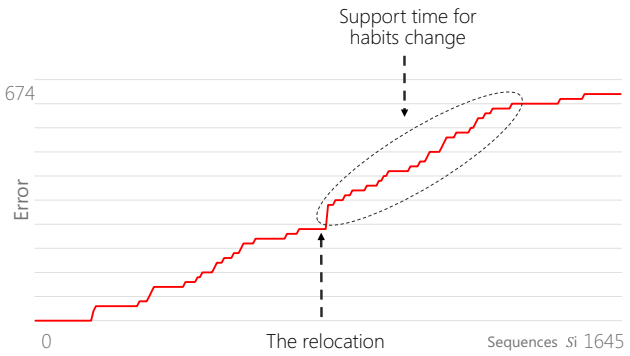

Fig. 9 Error evolution in DB-tree algorithm

Discussion Globally, our algorithm shows an interesting precision result with 70.5% of precision contrary to DB-tree and CVFDT which show a low rate with respectively 57 and 61.5% of precision.

To analyze the result presented in Table 2, we tracked the error evolution of the three algorithms (see Figs. 8, 9, 10).

After the relocation, our algorithm shows a strength to these shifts (precision decreases only from 72 to 69%) and support time for habits change is small because the algorithm detects a change in the user's habits and starts to add a supplement weight (drift's weight wd_j) until that the new sequences become frequents (see Fig. 8).

Contrariwise, DB-tree encounters difficulties to revive its model after the relocation to detect the new behaviors

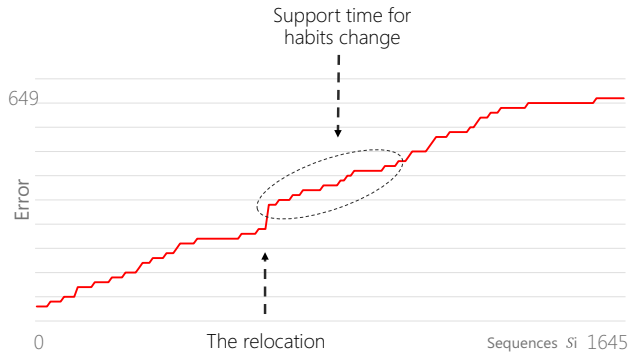

Fig. 10 Error evolution in CVFDT algorithm

(see the drop of the precision from 63 to 51% in Table 2). Indeed, as DB-tree traits all the sequences with the same manner (adds 1 to the concerned nods in the tree), it will take too much time to the new habits to become frequents (the minimum support will be increased by the old habits), which explains the high support time for habits change in Fig. 9.

CVFDT behaves better than DB-tree, but its global accuracy still considered as low (60.25%), back to the fact that theoretically CVFDT needs a massive set of examples to start improving its accuracy (the literature has mentioned a threshold of 100 k examples).

Despite the overall low accuracy, CVFDT seems unaffected by the change of habits, the recorded support time for habits change is smaller than DB-tree's one (see Fig. 10). This is justified by the fact that when CFDT detects a concept drift, it starts to build an alternate sub-trees using the new habits, these alternate sub-trees will replace the original ones when the error in the new sub-tree is less than the original error. This time needed to do this substitution is represented by our variable called support time for habits change.

After the experimentation of our approach in terms of precision and support of concept drift, we are going to test in the next section the computational impact of algorithm on the mobile resources.

4.5 Experimentation of mobile resources use

This work can be used in any environment (mobile, desktop or web applications) and using any architecture (local or distributed design), in spite of that, we are going to test our solution in a mobile environment, principally for these reasons: (1) users movements are usually collected incrementally using a mobile device, it is more consistent to continue predicting incrementally the users' movements on the same device. (2) Mobile environment requires careful handling of the reduced storage and computing capacities,

Table 3 Comparing our solution to Waze application in terms of memory usage

	Our solution	Waze
Memory usage (Mo)	40	67

if we prove that our solution is optimal for the mobile environment, it is clear that it will be useful for the other environments that have fewer requirements.

We tested our algorithm using an Android smartphone from Sony (Sony Xperia S) with 1 GB of Ram and 1.5 GHz dual-core processor.

The first test concerns the RAM usage, we added our solution to our precedent work (Boukhechba et al. 2015) where we recognized incrementally users' activities, then we compared the set with a well-known GIS solutions "Waze Social GPS Maps & Traffic", one of the best free navigation applications that won the best overall mobile app award at the 2013 Mobile World Congress, the reason of such selection is that Waze has a lot in common with our approach. In fact it gathers complementary map data and traffic information from its users like police traps (can be seen as a POI in our case), and learns from users' driving times to provide routing and real-time traffic updates.

Results in Table 3 that represent the average consumption of mobile's memory of every application for 12 h show that our solution is not greedy regarding memory usage with 40 Mo of RAM usage [note that the activity recognition system alone uses 34 Mo (Boukhechba et al. 2015)] compared to Waze with 67 Mo.

The second test concerns the storage capacity usage; we had to compare our solution with two algorithms: (1) DB-tree that uses the same tree structure as us, but that stores in descending order of support all items in the database; (2) Apriori algorithm that uses a traditional database structure. We chose Apriori to observe the difference between standard and tree databases. We tracked the variation of the database size in every solution in function of the number of sequences arrived (from 1 to 1 million sequences). In order to get such important number of sequences, we created an algorithm that generated random sequences containing between 2 and 20 POIs using 500 different POIs. Results exposed in Fig. 11 show how much the use of tree structure is benefic to the size of the database, thanks to sharing paths between items in the tree structure, our database had much smaller size (367 Mo) than in a traditional database (1200 Mo).

On the other hand, the maximum size of our tree (367 Mo) that is reached using one million sequences (if we take an average of two sequences per day, it represents more than 1388 years) represents a size widely acceptable by the requirements of mobile environment storage.

DB-tree had a slightly smaller database (250 Mo for one million sequences) than our algorithm, this is justified by the fact that DB-tree do not take into consideration the order of items, which means that "home, work, restaurant" and "restaurant, work, home" will be stored in the same branches. However, this technique can't be used when analyzing users' habits, because the order of habits is a very important parameter, otherwise, it will lead to gross errors.

Though, when comparing our algorithm to DB-tree in the real world, the difference will be neglected since the number of sequences will be much lower, for instance, from the experiment presented in Fig. 9, the average database size for our algorithm will be 260 Ko/year, for DB-Tree it will be 180 Ko/year. If the two algorithms will continue running for 10 years, the databases size will be 2.6 and 1.8 Mo respectively, the difference is too small to be considered.

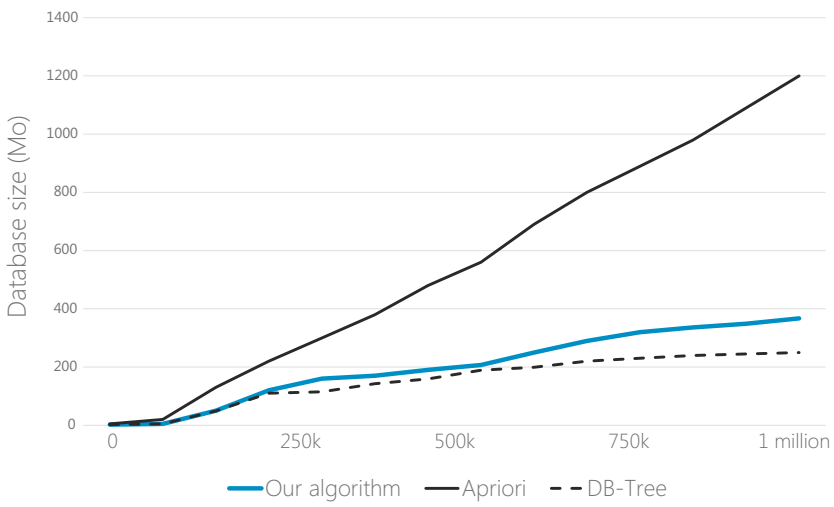

Fig. 11 Database size comparison between our solution and Apriori algorithm

5 Conclusions and future work

In this paper, we proposed a new algorithm based on the online learning of users' habits to predict users' next locations taking into account the changes that can occur in their routines. Our original contribution includes a new algorithm of online mining of association rules that support the concept drift.

Our approach has been experimented in a real case study using Geolife project to test the accuracy of our predicting technique. We compared our approach to a set of algorithms like Apriori, CVFDT and FP-growth algorithms via several assessments such as the ability to support users' habit changes and mobile resource usage. Results show that our proposal is well positioned compared to its similar, and represents an interesting solution to predict users' next activities without depleting the resources of users' mobile devices.

Several promising directions for future works exist. First, if this work is used in a big data context, some efforts shall be done to optimize the construction and the research process in the tree structure in order to minimize the response time of our algorithm. Secondly, clustering users' profiles represents an interesting research field, in that direction, the habits' tree represents a good structure that summarizes users' routines. Clustering users' profiles basing on their habits will be reduced to the comparison of two trees (habits' trees). Thirdly, this work has to be improved by introducing a temporal dimension to the habits' tree in order to improve our algorithm precision, for example, routines made in weekends are different of those made in working days.

References

Asahara A, Maruyama K, Sato A, Seto K (2011) Pedestrian-movement prediction based on mixed Markov-chain model. ACM Press, Chicago, Illinois, p 25. doi:10.1145/2093973.2093979

Boukhechba M, Bouzouane A, Bouchard B, Charles, GV, Sylvain G (2015) Online recognition of people's activities from raw GPS data: semantic trajectory data analysis. In: Presented at the 8th ACM international conference on pervasive technologies related to assistive environments

Boukhechba M, Bouzouane A, Bouchard B, Gouin-Vallerand C, Giroux S (2016) Energy optimization for outdoor activity recognition. J Sens 2016:1–15. doi:10.1155/2016/6156914

Ezeife CI, Su Y (2002) Mining incremental association rules with generalized FP-tree. In: Cohen R, Spencer B. (eds) Advances in artificial intelligence. Springer, Berlin, pp 147–160

Gai YL (2012) Research on data mining and apriori algorithm. Adv Mater Res 546–547:497–502. doi:10.4028/www.scientific.net/AMR.546-547.497

Gama J, Medas P, Castillo G, Rodrigues P (2004) Learning with drift detection. In: Bazzan A.L.C., Labidi S (eds) Advances in artificial intelligence—SBIA 2004. Springer, Berlin, pp 286–295

Gama J, Sebastião R, Rodrigues PP (2009) Issues in evaluation of stream learning algorithms. ACM Press. doi:10.1145/1557019.1557060

Gambs S, Killijian M-O, del Prado Cortez MN (2012) Next place prediction using mobility Markov chains. ACM Press, pp 1–6. doi:10.1145/2181196.2181199

Hipp J, Güntzer U, Nakhaeizadeh G (2000) Algorithms for association rule mining—a general survey and comparison. ACM SIGKDD Explor Newsl 2:58–64. doi:10.1145/360402.360421

Kapterev A (2014) Where to go next. In: Presentation secrets. Wiley, Hoboken, pp 265–278

Katsaros D, Nanopoulos A, Karakaya M, Yavas G, Ulusoy Ö, Manolopoulos Y (2003) Clustering mobile trajectories for resource allocation in mobile environments. In: Berthold RM, Lenz H-J, Bradley E, Kruse R, Borgelt C (eds) Advances in intelligent data analysis V. Springer, Berlin, pp 319–329

Li K, Fu Y (2014) Prediction of human activity by discovering temporal sequence patterns. IEEE Trans Pattern Anal Mach Intell 36:1644–1657. doi:10.1109/TPAMI.2013.2297321

Morzy M (2006) Prediction of moving object location based on frequent trajectories. In: Levi A, Savaş E, Yenigün H, Balcısoy S, Saygın Y. (eds) Computer and information sciences—ISCIS. Springer, Berlin, pp 583–592

PhridviRaj MSB, GuruRao CV (2014) Data mining—past, present and future—a typical survey on data streams. Proced Technol 12:255–263. doi:10.1016/j.protcy.2013.12.483

Simmons R, Browning B, Zhang Y, Sadekar V (2006) Learning to predict driver route and destination intent. IEEE, pp 127–132. doi:10.1109/ITSC.2006.1706730

Spaccapietra S, Parent C, Damiani ML, de Macedo JA, Porto F, Vangenot C (2008) A conceptual view on trajectories. Data Knowl Eng 65:126–146. doi:10.1016/j.datak.2007.10.008

Witten IH, Frank E, Hall MA (2017) Data mining: practical machine learning tools and techniques. Morgan Kaufman, 654 p

Zhang X, Germain C, Sebag M, 2010. Adaptively detecting changes in Autonomic Grid Computing. IEEE, pp 387–392. doi:10.1109/GRID.2010.5698017

Zheng Y, Wang L, Zhang R, Xie X, Ma W-Y, 2008. GeoLife: managing and understanding your past life over maps. IEEE, pp. 211–212. doi:10.1109/MDM.2008.20