Online Prediction of People's Next Point-of-Interest: Concept Drift Support

Conference Paper · September 2015

DOI: 10.1007/978-3-319-24195-1_8

5 authors, including:

Abdenour Bouzouane
University of Québec in Chicoutimi
99 PUBLICATIONS **367** CITATIONS
SEE PROFILE

Bruno Bouchard
108 PUBLICATIONS **369** CITATIONS
SEE PROFILE

Charles Gouin-Vallerand
Télé-université
26 PUBLICATIONS **93** CITATIONS
SEE PROFILE

Sylvain Giroux
Université de Sherbrooke
153 PUBLICATIONS **752** CITATIONS
SEE PROFILE

Online prediction of people's next Point-of-Interest: Concept drift support

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

[1]LIARA Laboratory University of Quebec at Chicoutimi (UQAC)
Chicoutimi, G7H 2B1, Canada
{Mahdi.Boukhechba1, Abdenour_Bouzouane,
Bruno_Bouchard}@uqac.ca
[2]Tele-universite of Quebec (TELUQ)
charles.gouin-vallerand@teluq.ca
[3] University of Sherbrooke
sylvain.giroux@usherbrooke.ca

Abstract. Current advances in location tracking technology provide exceptional amount of data about the users' movements. The volume of geospatial data collected from moving users' challenges human ability to analyze the stream of input data. Therefore, new methods for online mining of moving object data are required. One of the popular approach available for moving objects is the prediction of the unknown future location of an object. In this paper we present a new method for online prediction of users' next important locations to be visited 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 of online mining association rules that support the concept drift.

Keywords: Mobile environment, Human activities, Activity prediction, Online association rules, Spatio-temporal data mining, Concept drift.

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 kind of moving objects, generating huge volumes of mobility data. Despite the fact that data collected from mobile devices is more accurate, there is still scientific locks regarding the use of this data. One of these important research topics is the prediction of user's next location [5, 7, 16], where results of these research are used in a wide range of fields like traffic management, public transportation, assistance of people with special needs, commercials and advertising.

Our precedent work [2] proposed an online activity recognition system that offers the possibility to understand what people are doing at a specific moment by inferring incrementally people's interesting 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 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. Assuming that we have some beforehand knowledge about people's destinations, our precedent work was able to detect anomaly in the users' behaviors (comparing the planned and the real destination). The predictive model that we are proposing in this paper aims to launch assistance processes when users are lost 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' routine is the concept drift [19, 20, 21]. 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 "@1" for one year, after that he moves to "@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 a user is doing a new frequent habits (if we take a support of 60 % using Apriori[1] algorithm [15] for example, it will take 7 months to detect the new habits), in the meantime, all what is proposed by these algorithms is probably false since it is based on the old routines and not the new ones.

Current work [16, 1, 7, 8, 11] that try to learn users' routines and predict their future routines fail in their ability to deal with the changes in users' behaviors during the learning process, and there are only few works 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 new habits detected to become quickly frequent, by introducing a new criterion of support calculation based on a weight distribution of data collected and not the classic number of occurrences.

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

[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 Related work

Significant research effort has been undertaken in both mobile computing and spatial data mining domains [17, 18]. Many advances in tracking users' movements have emerged resulting in several proposals for predicting future users' locations. The main approach proposed is to learn the user's patterns from his historical locations and try to predict the next location via different techniques.

In [8] 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 and 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, these works do not permit incremental training of the models, since it is based on a posteriori learning, secondly, the fact that authors propose matching functions in form of strategies (simple, polynomial, logarithmic and aggregation strategies) can create a computational complexities; difficulties to choose and set the rights 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 [7]. 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. proposed in [1] 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 an accuracy of 74.4%

[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.

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 [3] present two new algorithms that use the frequent patterns tree (FP-tree) 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, as well as counts all items in all transactions in the database in its branches. The DB-Tree is constructed 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 itemsets to reduce the number of times the original database needs to be scanned when previous small itemsets 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 [10], 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 [11]. 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' habits changes, since there is no approach that handle the habits' drift during the learning and the prediction of next location process.

3 OVERVIEW OF THE APPROACH

Assuming that we track a user outdoor activities using an incremental solution such as that one presented in [2], every activity (location) detected is called a Place of Interest (POI). A POI is an urban geo-referenced object where a person may carry out a specific activity. Our approach begins by constructing a sequence of POI_j 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. Finally, the algorithm predicts the next POI using the association rules drawn from HT.

3.1 Sequence construction

This step aims to represent users' habits via monitoring of routines called sequence S_i, every sequence contains a set of disjoint singletons POI_j (c-e we can't find the same POI many times in the same sequence) and terminates with 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 S1 and creates a new sequence S2, the reason of our proceeding is to optimize the storage of the sequences.

Algorithm 1: Sequence construction

```
Input:
A POI_j;
Output:
Sequence S_i;
S_i = null ;
For each new POI_j
   If (! S_i.contains(POI_j) and  StillTheSameDay) then
   S_i = S_i + POI_j
   Else //new sequence
      Return S_i;
      If (StillTheSameDay) S_i = Last POI_j
      Else S_i = null ;
End for each
```

Algorithm 1 is executed on every POI_j arrival, when S_i 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, this improvement is based on the treatment and the storage of each POI_j in HT when it arrives without waiting for the construction of S_i.

3.2 Habits' tree update

Habits' tree HT is a data structure that takes form of a special tree, every node represents a POI and is characterized by a weight w_j that represents the weight of POI_j's occurrence and an identifier ID_j that aims to identify the sequences by an integer (see 4.3.1 Sequence identification).
The work of [3] 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 [22]) which stores in descending order of support all items in the database, as well as counts all items in all transactions in the database in its branches. The DB-Tree is constructed 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 wasn't chosen arbitrary, using this structure and storing all the sequences (frequents and not frequents) 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 figure 2).
To give a brief idea on the way the data are structured in HT, we illustrated in figure 1 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 >.

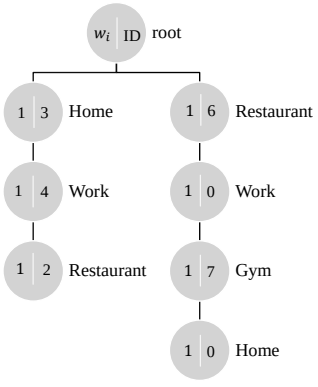

Fig.1. HT Structure construction

We added two new sequences S_3 = Home, Work, Gym, Cinema and S_4 = Home, Gym, from the updated HT presented in figure 2, we can observe how the notion of sharing paths (in the nodes home and work) leads to a compression of the database dimension. Additionally, we can notice that the algorithm added these sequences with different weight than the previous because it detects new behaviors, more details are provided below.

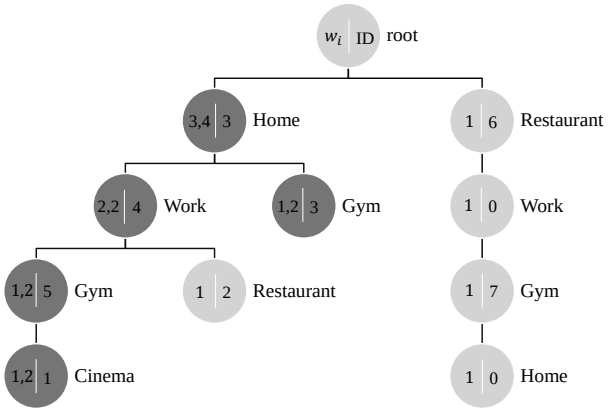

Fig.1. Sharing paths in HT structure

On the arrival of a new sequence S_i , the algorithm 2 recursively processes each POI_j in S_i . If the POI_j exists in HT, the concerned node's weight is updated, otherwise, the algorithm adds a new node with a new random POI_j. ID, and a new weight $POI_j.w$ where the details of calculation is given in the next section. For example, supposing that after a certain time of learning, user's HT is structured like on figure 2, the next day, the user did the following sequence: home, work, gym, @1. Figure 3 shows how our algorithm updates the nodes: home, work, gym; and adds a new POI: @1. Note that @1 was added to HT with an unknown weight because it will be calculated in the next section.

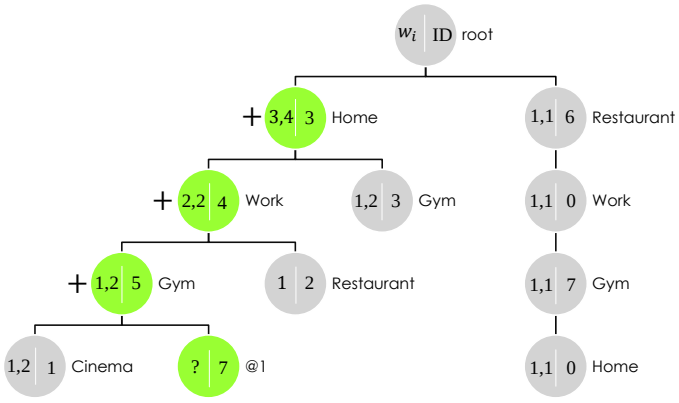

Fig.2. Example of HT updated

Algorithm 2: Habits' tree update

```
Input:
A sequence S_i;
Output:
HT;
ArrayListPOI = S_i.ToList() ;
For (int i=0 ; i< ArrayListPOI.Lenght; i++)
New Node = ArrayListPOI [i] ;
If (HT.contains (NewNode))
HT.UpdateNode (NewNode)  ;
HT.GoDown;
Else
// Add the remaining POI_j in S_i and leave the loop
Ht.AddAllNodes (from ArrayListPOI [i] to ArrayListPOI
[last]);
Break;
And if
End for
```

The algorithm 2 shows how to maintain the HT on every new S_i arrival, but we haven't yet responded to our problem requirements. The next section presents how we proceed to distribute the weight in every node updated in HT. Our contribution aims to track the drift in users' habits, and to distribute the weights basing on these behavior changes.

3.3 Drift detection

This part aims to track the changes in users' habits. Our technique is divided into two steps: firstly we formulate mathematically each new sequence and secondly we use this number to test if 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. The introduction of ID_j in each node was in this perspective, in fact, every new sequence will be represented by a variable called x_j where x_j is obtained from the concatenation of each POI_j.ID present in S_i. For instance, in the example cited earlier, the new arrived sequence is: Home, work, gym, @1; using each node's ID that corresponds to each POI_j (see Figure 3), the variable x_j will take a value of 3457. 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) [12] to detect the 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 [19]. This test considers a cumulative variable U_T defined as the cumulated difference between the observed values (in our case the sequences' identifier x_j) and their mean till the current moment.

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

$$\begin{cases} U_t = \sum_{j=1}^{t} \left(x_j - \overline{x_t} - \delta\right), U_0 = 0 \\ m_t = \min(U_t), t \geq 1 \\ PHT = U_t - m_t \end{cases}$$

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

$$\begin{cases} U_t = \sum_{j=1}^{t} \left(x_j - \overline{x_t} + \delta\right), U_0 = 0 \\ M_t = \max(U_t), t \geq 1 \\ PHT = M_t - U_t \end{cases}$$

Where $\overline{x_t} = \frac{\left(\Sigma_{j=1}^{t} x_j\right)}{t}$ 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 in [14], where authors propose a self-adaptive method of change detection by proving that λ_t can be self-adapted using this equation: $\lambda_t = f * \overline{x_t}$ 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 usually doesn't change his habits gradually, so, we don't need a big number of witnesses to detect the changes, that's why we put $f = 2$.

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 Weight distribution

On every S_i arrival, we calculate the weight w_j that will be added to every node concerned by the new sequence in HT, the distribution is realized using a variable called drift's weight (wd_j) calculated using an exponential function like

follows: $wd_j = 1 - e^{-\frac{1}{2}\frac{PHT}{\lambda_t}}$, 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, from figure 4, when $p_t = 0$ because PHT = 0, we are sure that the user is doing the same habits, by against, each time p_t approaches the value 1, or exceeds it, we conclude that the user has a drift in his habits.

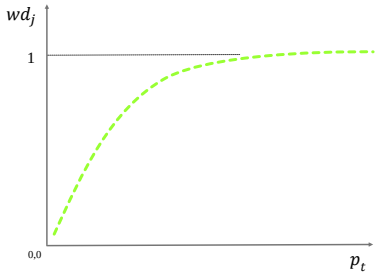

Fig.3. Mathematical representation of wd_j

Consequently, the weight added to HT's concerned nodes is like follows:

$$\begin{cases} w_j = 1 + wd_j \text{ , } if S_i \text{ is not } frequent \\ w_j = 1 \qquad\qquad \text{, } if\ S_i \text{ is } frequent \end{cases}$$

As we see, our distribution behaves in two ways (see algorithm 3): 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_j), the reason why we proceed this way is that we are trying to help only the new habits that aren't frequents to become frequents, once arrived, we stop our help to not promote a sequence (habit) relative to another.

Algorithm 3: Weight distribution

```
Input:
A sequence S_i;
Output:
Weight w_j ;
x_j = HT.GetSequenceId(S_i);
If (S_i.isfrequent()) w_j = 1 ;
Else
      Calculate λ_t, U_T, m_t, M_t;
      Calculate PHT;
      Calculate wd_j;
      w_j = 1 +wd_j ;
End if
Return w_j
```

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 Activity prediction

Association rules mining. Our technique is inspired from FP-growth algorithm and one of its incremental versions called DB-Tree [3], the main difference between our work and theirs is that we introduced a weight distribution function that tracks the habits drift, secondly, our structure doesn't order the tree's items in descending order of support like done in DB-tree, in fact, DB-tree and FP-growth don't make a difference between "home, work, restaurant" and "restaurant, work, home", clearly, we can't 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 figure 2, I = {Home, gym, work, @1, restaurant, cinema} and MinSupport = 60% of database transactions. To compute the frequent POI_j after constructing the habits tree HT, the algorithm mines the frequent POI_j that satisfy the minimum support represented by the MinSupport percentage of the maximum item's weight in HT. From figure 2, we have the weight of every items like follows (note that for every item we add up the corresponding weights found in all the tree, for example Home's weight = 3.4 + 1.0): 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_j are all items greater or equal to 2.64 as {(Home: 4.4), (gym: 3.4), (work: 3.2)}.

The next steps is mining the frequent patterns from HT and association rules that are quite similar to those in FP-growth [3], 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 (figure 2) these rules: {work, gym->home// work ->gym// home->gym// home-> work// Home, work->gym}, predicting the next activity lies on choosing the most appropriate association rules that represent user's 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 we can predict that he will go next to the gym. Unfortunately, it's not always so simple to choose the right association rules. For instance, if we noticed that the user starts from his home and we try to predict the next activity, we will find that there are two rules that start from home, one predicts gym as next localization, and the other predicts work, and both are rights because the user has the habit of going sometimes to the gym and sometimes to work from home. This kind of conflict is managed by our algorithm (see algorithm 4) by choosing the rule with the greater weight. Clearly, this technique doesn't rule out the risk of errors (this is what explain the error percentage presented in the experimental evaluation section) but minimizes it since we choose the most recurrent rule.

Algorithm 4: Activity prediction

```
Input:
Habits tree HT;
Users past activities UPA;
Output:
Next activity;
TheRule = null;
ListRules = HT.GetAssociationRules();
ListAppropriateRules=listRules.GetAllThatContains(UPA);
If (ListAppropriateRules.size >1)
TheRule = ListAppropriateRules.GetRuleWithMaxWeight();
Else
TheRule = ListAppropriateRules.get(0);
End if
Return TheRule.getResultingClause();
```

Finally, we present in algorithm 5 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 5, 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’s not expensive in term of time 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 (user make mistakes because he doesn’t know what to do next), at this time the system searches for the association rules and predict the next activity.

Algorithm 5: final algorithm

Input:
A POI_j;
Users past activities UPA
Output:
Next activity;
S_i = Sequence construction (POI_j);
w_j = Weight distribution (S_i) ;
HT = Habits 'tree update(S_i);
Next location = Activity prediction (HT,UPA)
Return Next location ;

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 to three users with different profiles to note their daily habits for three months, the choosing of users wasn't 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.
Real data. To push even further the level of our experiment, we used a renowned dataset from the Microsoft research project GeoLife [4]. The GPS trajectory dataset was collected in (Microsoft Research Asia) Geolife project by 182 users in a period of over three 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+ hours. 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 seconds or every 5~10 meters 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 entertainments and sports activities, such as shopping, sightseeing, dining, hiking, and cycling.
We treated the dataset to add some information to the database like user's speed and orientation to be used in our previous work [2] to detect users' activities from their GPS traces, every POI recognized was identified by its unique combination (longitude latitude), the result was organized in dataset (see table 1) as user's daily routines.

Table 1 : Example of the inferred POIs dataset from GeoLife

ID	Longitude	Latitude	Name	Type	Date
1	116.3216	39.99190	Building 13	House	2008-10-23-02-53-30
2	116.3214	40.01119	Swissotel Beijing	hotel	2008-10-23-04-15-52
…	…	…	…	…	…

4.2 Testing process

Test processes in association rules represent a delicate step, since how to test is related to the field of study. We created a testing process specific to us 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 user's movements but in every sequence of POI, the virtual user will forget his next destination, in this case our algorithm will predict a next localization that will be compared with the real next activity (see figure 5).

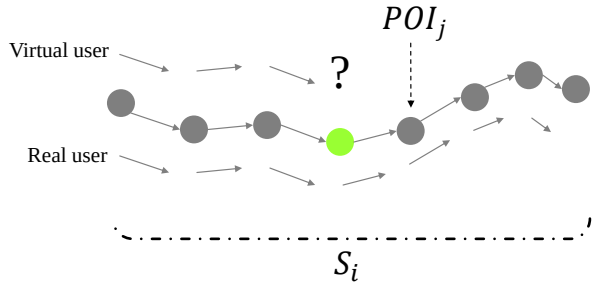

Fig.4. Testing process with real and virtual user

The position of the virtual user's moment of forgetting 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's the most recurrent activity after home. All the times when the user will go to drive 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 2 represent the precision, GE, LE and HE of our algorithm on every user data.

Table 2: Standard incremental activity prediction results

	Simulated data			Geolife
	User 1	User 2	User 3	User 4
Precision	83 %	71 %	61 %	68 %
GE	17 %	29 %	39 %	32 %
LE	11 %	14 %	13 %	9 %
HE	6 %	15 %	26 %	23 %

From table 2, 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 shows a precision of 68% and a global error of 32%.
Discussion. First, it is clear that our approach shows an interesting result with an average precision of 70.75%, moreover, analyzing the distribution of errors in every user's data, we notice a strong correlation between the global error GE and the habits 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, so, regularity in users' movements is important to succeed in predicting his next location.

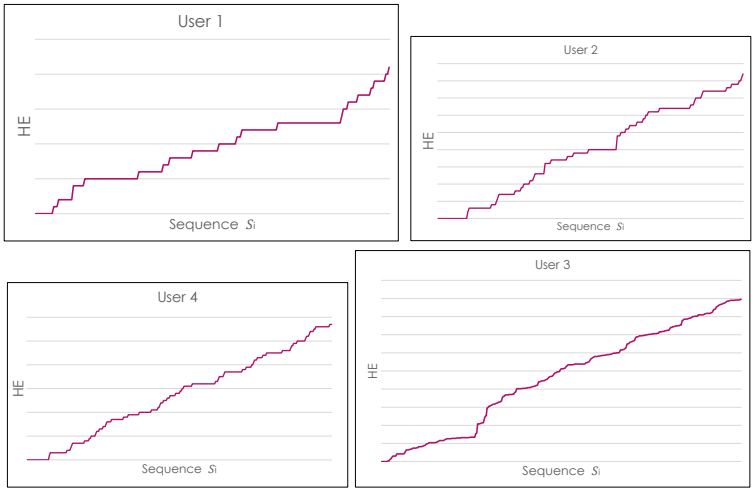

Fig.5. Habit error HE evolution in the four users' data

The figure 6 tracks 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 predict 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 explain the continuous increase of HE over time.

4.4 Incremental activity prediction with concept drift experiment

We compare in this section our approach with an incremental version of FP-growth called DB-Tree [3], the ideal dataset to experiment the two 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.

Table 3 : Comparison between our algorithm and DB-tree algorithm

	Our algorithm	DB-tree algorithm
Precision before the relocation	72 %	63 %
Precision after the relocation	69 %	51 %
Global precision	70.5 %	57 %

Table 3 presents the results of our experiment; we divided it into three indicators, precision before and after the relocation, and the global precision.

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

First, the reason why our algorithm shows better results (72% against 63% for DB-tree) in normal dataset (before the relocation) is that FP-tree doesn't handle the order of POIs in sequences since it reorders the POIs to store them in descending order of support in the tree, therefore, for example, the habits of going from home to work and from work to home are the same for FP-tree, which is quite wrong.

To analyze the after location step, we tracked the error evolution of both algorithms (see figure 7 and 8) to justify the results shown in table 2.

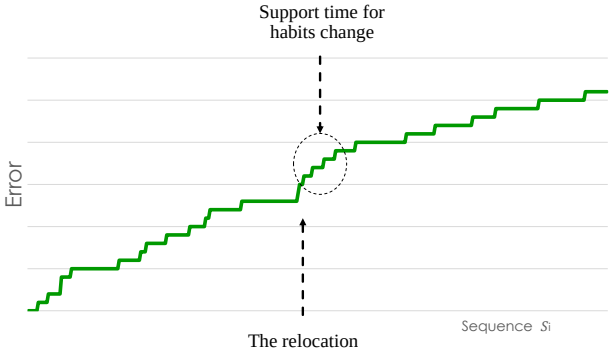

Fig.6. Error evolution in our algorithm

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.

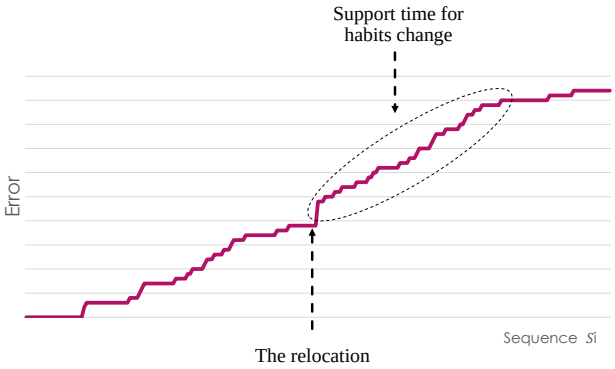

Fig.7. Error evolution in DB-tree algorithm

Contrariwise, DB-tree encounters difficulties to revive its model after the relocation to detect the new behaviors (drop of the precision from 63% to 51%). Indeed, as DB-tree traits all the sequences with the same manner (adds 1 to the concerned nods in the tree), it will take much time to the new habits to become frequents (the minimum support will be increased by the old habits), what explains the important support time for habits change in figure 8.
After the experimentation of our approach in term 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, 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 less 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 [2] 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 4 that represent the average consumption of mobile's memory of every application during 12 hours 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 [2]) comparing to Waze with 67 Mo.

Table 4: Comparing our solution to Waze application in term of memory usage

	Our solution	Waze
Memory usage	40 Mo	67 Mo

The second test concerns the storage capacity usage; we had to compare our solution with an algorithm that uses a traditional database structure to observe the impact of using a tree structure. We compared our approach with the Apriori algorithm, a widely used algorithm for association rule learning using a standard database design. 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 a such important number of sequences, we created an algorithm that generated random sequences containing between 2 to 20 POI using 500 different POIs. Results exposed in figure 9 show how much the use of tree structure is benefic to the size of the database, thanks to sharing paths between items in tree structure which leads to much smaller size than in a traditional database.

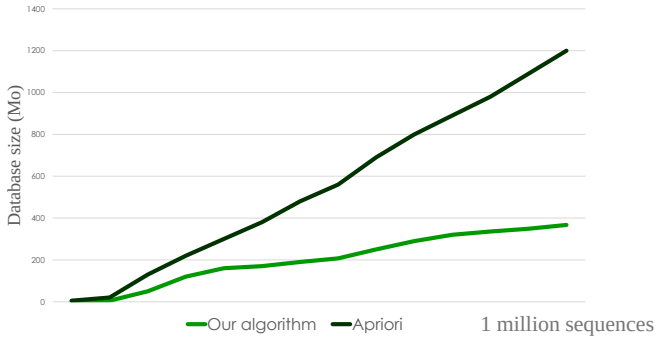

Fig.8. Database size comparison between our solution and Apriori algorithm

In the other hand, the maximum size of our tree (467 Mo) that is reached using one million sequences (if we take an average of 2 sequences per day, it represents more than 1388 years) represents a size widely acceptable by the requirements of mobile environment storage.

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 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 by comparing it to Apriori algorithm and DB-tree algorithm via several indicators like supporting users' habits changes and mobile resources 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 to minimize the response time of our algorithm. Secondly, the clustering of 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).

6 References

[1] Asahara, A., Maruyama, K., Sato, A. , Seto, K.: Pedestrian-movement prediction based on mixed Markov-chain model. Proc. The 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM (2011), 25–33.

[2] Boukhechba, M., Bouzouane, A., Bouchard, B., Gouin-Vallerand, C., Giroux, S.: Online recognition of people's activities from raw GPS data: Semantic Trajectory Data Analysis. Proc. The 8th ACM International Conference on Pervasive Technologies Related to Assistive Environments, July 2015.

[3] Ezeife, C. I. and Su, Y.: Mining Incremental Association Rules with Generalized FP-Tree. Proc. The 15th Canadian Conference on Artificial Intelligence, May 2002.

[4] Zheng, Y., Xie, X., Ma, W.: GeoLife: A Collaborative Social Networking Service among User, location and trajectory. Invited paper, in IEEE Data Engineering Bulletin. 33, 2, 2010, 32-40.

[5] Zheng, Y., Zhang, L., Xie, X., Ma, W.: Mining interesting locations and travel sequences from GPS trajectories. In Proceedings of International conference on World Wild Web (WWW 2009), Madrid Spain. ACM Press: 791-800.

[6] Zheng, Y., Li, Q., Chen, Y., Xie, X., Ma, W.: Understanding Mobility Based on GPS Data. In Proceedings of ACM conference on Ubiquitous Computing (UbiComp 2008), Seoul, Korea. ACM Press: 312-321.

[7] Gambs, S., Killijian, M., Cortez, D. P., Miguel, N.: Next place prediction using mobility Markov chains. Proc. The 1st Workshop on Measurement, Privacy, and Mobility, ACM (2012), 1–6.

[8] M. Morzy.: Prediction of moving object location based on frequent trajectories. ISCIS, volume 4263 of *LNCS*, pages 583–592. Springer, 2006.

[9] M. Morzy.: Mining frequent trajectories of moving objects for location prediction. MLDM, volume 4571 of *LNCS*, pages 667–680. Springer, 2007.

[10]Katsaros, D., Nanopoulos, A., Karakaya, M, Yavas, G., Ulusoy, O., Manolopoulos, Y.: Clustering mobile trajectories for resource allocation in mobile environments, in: Intelligent Data Analysis Conference (IDA2003)Lecture Notes in Computer Science, vol. 2810, Springer-Verlag, 2003.
[11] Liu, T., Bahl, P., Chlamtac, I.: Mobility modeling, location tracking, and trajectory prediction in wireless ATM networks, IEEE J. Select. Area Commun. 16 (6) (1998) 922–936.
[12] E. S. Page. Continuous Inspection Schemes. Biometrika, Vol. 41:100-115, 1954
[13] Mouss, H., Mouss, D., Mouss, N., Sefouhi, L.: Test of Page-Hinkley, an Approach for Fault Detection in an Agro-Alimentary Production System. Proceedings of the 5th Asian Control Conference, Vol.2, pages: 815-818, 2004.
[14] Zhang, X., Germain-Renaud, C., Sebag, M.: Adaptively Detecting Changes in Autonomic Grid Computing. Proc. The 11th ACM/IEEE International Conference on Grid Computing (Grid 2010)(Autonomic Computational Science Workshop), 387-392.
[15] Agrawal, R., Srikant, R..: Fast algorithms for mining association rules in large databases. Proceedings of the 20th International Conference on Very Large Data Bases, VLDB, pages 487-499, Santiago, Chile, September 1994.
[16] Cheng, C., Yang, H., Lyu, M. R., King, I..: Where You Like to Go Next: Successive Point-of-Interest Recommendation, Proceedings of the 23 International Joint Conference on Artificial Intelligence, August 3-9-2013, Beijing, china pp 2605-2611(2013).
[17] Spaccapietra, S., Parent, C., Damiani, M.L, Macêdo, J., Porto, F., Vangenot, C.:. A conceptual view on trajectories. *Data Knowl. Eng*. 65(1): 126-146 (2008).
[18] Zheng, Y., et al. GeoLife: Managing and understanding your past life over maps. In *Proceedings of MDM'09*, (Beijing China, April 2008), IEEE Press: 211-212.
[19] Gama, J., Sebasti˜ao, R., Rodrigues, P.P.: Issues in evaluation of stream learning algorithms. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2009). pp. 329–337. ACM Press, Paris, France (2009).
[20] Gama, J., Medas, P., Castillo, G., and Rodrigues, P.: Learning with drift detection. In Ana L. C. Bazzan and Soane Labidi, editors, Advances in Artcial Intelligence – SBIA 2004, volume 3171 of Lecture Notes in Computer Science, pages 286{295. Springer Verlag, October 2004.
[21] Kuncheva L.I.: Classifier ensembles for detecting concept change in streaming data: Overview and perspectives, Proc. 2nd Workshop SUEMA 2008 (ECAI 2008), Patras, Greece, 2008.
[22] Hipp, J.; Güntzer, U.; Nakhaeizadeh, G.: Algorithms for association rule mining: a general survey and comparison. ACM SIGKDD Explorations Newsletter 2: 58.