Sequential Patterns for Spatio-Temporal Traffic Prediction

Feda Almuhisen, Nicolas Durand, Leonardo Brenner, Mohamed Quafafou

Sequential Patterns for Spatio-Temporal Traffic Prediction

Feda Almuhisen
Aix Marseille Univ, Université de Toulon, CNRS, LIS
Marseille, France

Nicolas Durand
Aix Marseille Univ, Université de Toulon, CNRS, LIS
Marseille, France

Leonardo Brenner
Aix Marseille Univ, Université de Toulon, CNRS, LIS
Marseille, France

Mohamed Quafafou
Aix Marseille Univ, Université de Toulon, CNRS, LIS
Marseille, France

ABSTRACT

This paper proposes a method for predicting the traffic status of a city within time windows. The method takes advantage of space-partitioning, closed sequential pattern extraction, emerging pattern detection, and Markov chain modeling. From trajectories, we identify active regions in which moving objects mostly visit. The traffic status of each region is detected based on continuous tracking of closed sequential patterns evolution over time. Based on the proposed Markov model, the near-future status of traffic is then predicted. The traffic status is reported on maps and can be used to enhance future city transportation. The experiments on real-world data sets show that the proposed method provides promising results.

KEYWORDS

Closed sequential patterns, emerging patterns, Markov chains, moving object trajectory data, traffic prediction and monitoring

1 INTRODUCTION

Smart city concept has become essential to manage the city and to provide innovative services to citizens. Traffic prediction is one of these services and aims to improve the mobility in the city [16]. Indeed, the increasing of the urban population makes the creation of intelligent transportation systems strategic. IT platforms and sensor networks play a crucial role in the management of the spatio-temporal data, preprocessing, traffic prediction and services [6] [30]. Traffic prediction [25] refers to: (1) traffic status prediction: congestion prediction in future time, (2) traffic flow prediction: prediction of traffic flows in future time, and (3) travel demand prediction: prediction of people's travel demands in order to better dispatch taxi vehicles for different regions. The practical goal of this work is to deliver a global view of current or future traffic status in regions of interest (see Figure 6). Prior short term traffic predictions models can be used directly by experts to take relevant actions against prevailing or incipient traffic trends. Developing such prediction systems needs trajectory historical data analysis which can be also used in other applications such as improving route navigation and urban area planning [14] [17]. Markov models have been widely used in the field of traffic prediction in order to improve traffic flow [29], predict traffic conditions [20] [27], forecast travel speed [28], and predict routes [13].

In this paper, we focus on traffic status prediction by analysing the evolution of traffic. In fact, we analyze the trajectories of vehicles in historical data to detect and visualize the evolution of traffic in the city. We propose a new method for predicting the traffic status (increasing, decreasing, etc.) in the next future. This method is called SP4TP (Sequential Patterns for Traffic Prediction) and its originality is to leverage space partitioning, closed sequential pattern mining [9], emerging pattern mining [7], and Markov chain modeling [22]. The space area is divided into regions (uniformed squares) using a given spatial granularity value. Considering a selected region and a time windows, our method based on Markov models predicts the next traffic status. To train the proposed Markov model, we determine the traffic status by detecting the evolution types (Emerging, Decreasing, etc.) of sequential formal concepts (containing closed sequential patterns) extracted from trajectory data [2] [3]. The prediction results are visualized in geotagged maps. To the best of our knowledge, no prior methods based on sequential patterns, emerging patterns, and Markov models have been proposed to build traffic prediction models. Two real-world data sets from two cities, Beijing and San Francisco, are used in the experiments to evaluate our method and compare it to an itemset based approach and a baseline method which does not use patterns. We also assess the advantage of using sequential patterns in our context.

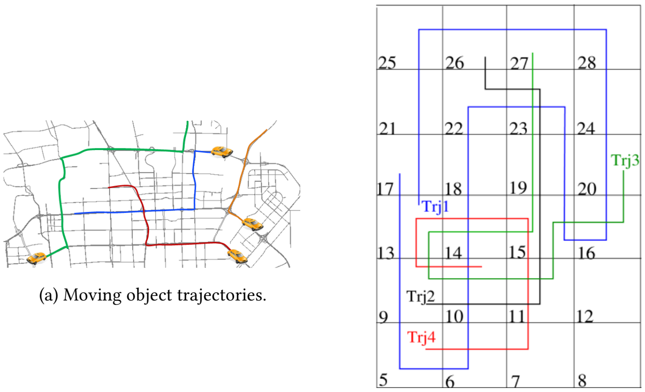
The rest of the paper is organised as follows. Section 2 presents the notions useful for understanding the paper. Related work is discussed in Section 3. The proposed method, SP4TP, is detailed in Section 4. Section 5 presents the experimental results. Finally, we conclude in Section 6.

2 PRELIMINARIES

In this section, we present the notions useful to understand the proposed method (in Section 4): frequent itemsets, sequential patterns, formal concepts, emerging patterns, and Markov chains. We illustrate some notions on an example: a set of moving object trajectories (see Figure 1 (a)).

2.1 Processing Trajectories to Sequences

A trajectory Trj_i is an ordered set of points. A point is a timestamped GPS-location, i.e., (time, latitude, longitude) [30]. Figure 1 (b) presents 4 trajectories over two days. The space area is divided into uniform grid cells, e.g., 40 meters (corresponding to the spatial granularity value). Each trajectory is split according to the time granularity value, e.g., one day, and labelled by the corresponding time value. Let us note that a trajectory can be also split into sub-trajectories reflecting several journeys in relation to the detected long stops, for instance. Trj_1 has been slit into 5 sequences: s_1, s_2, …, s_5. Trajectories are then mapped in the raster area to obtain the data set presented in Figure 1 (c). For instance, the sequence s_2 occurs in Day 1 and is composed of grid cells 20, 16 and 15.



(a) Moving object trajectories.

(b) Mapping on raster area.

	seq-Id	Sequences	Label
Trj_1	s_1	<(13)(17)(21)(25)(26)(27)(28)(24)>	Day1
	s_2	<(20)(16)(15)>	Day1
	s_3	<(19)(23)(22)(18)>	Day1
	s_4	<(14)(10)(6)>	Day1
	s_5	<(5)(9)(13)(17)>	Day1
Trj_2	s_6	<(9)(10)(11)(15)>	Day1
	s_7	<(19)(23)(22)(26)>	Day1
Trj_3	s_8	<(20)(16)>	Day2
	s_9	<(15)(11)(10)(9)(13)>	Day2
	s_{10}	<(14)(15)(19)(23)(27)>	Day2
Trj_4	s_{11}	<(5)(6)(7)(11)(15)>	Day2
	s_{12}	<(14)(13)(9)(10)>	Day2

(c) Sequential data set with time labels.

Figure 1: From trajectories to sequences.

2.2 Frequent Itemsets and Sequential Formal Concepts

Frequent itemsets have been introduced in [1]. Their extraction corresponds to finding the sets of items (i.e., attribute values) that appear simultaneously in at least a certain given number of transactions (i.e., objects) recorded in a database. In the case of ordered items in the data set (like trajectory data), sequential patterns are extracted [9].

Let $\mathcal{I} = \{i_1, ..., i_k\}$ be a set of items. A subset $I \subseteq \mathcal{I}$ is called an itemset. $|I|$ denotes the number of items of I. A sequence is an ordered list of itemsets. It can be represented as $s = <(I_1)(I_2)...(I_k)>$ where each I_i is a subset of $\mathcal{I}$ and I_i comes before I_j if $i \leq j$. The aim is to find all the frequent subsequences in a sequential database. Let $\mathcal{S} = <s_1, ..., s_n>$ be a list of sequences in a database. A sequence $s_1 = <a_1, \ldots, a_m>$ is a subsequence of another sequence $s_2 = <b_1, \ldots, b_n>$, denoted by ($s_1 \subseteq s_2$), if there are integers $1 \leq i_1 \leq i_2 \leq \cdots \leq i_m \leq n$, such as $a_1 \subseteq b_{i_1}$, $a_2 \subseteq b_{i2}$, …, and $a_m \subseteq b_{im}$. s_2 is called a super-sequence of s_1 (s_2 contains s_1). The support of a sequence s_n, noted $support(s)$, is the number of occurrences of s_n in $\mathcal{S}$. A sequence s is said to be a sequential pattern (also called a frequent sequence) if $support(s) \geq minsup$, where *minsup* is the minimum support threshold value set by the user. One of the limitations of using sequential pattern algorithms, is that it generates the full set of sequential patterns and scans the database many times when a longer pattern exists. This is why, closed sequential patterns are interesting. Closed sequential patterns are the set of sequential patterns that are not included in other sequential patterns having the same support because they represent the largest frequent subsequences common to sets of sequences. A sequential pattern s_1 is closed if $\nexists s_2$ with $s_1 \subset s_2$ and $support(s_1) = support(s_2)$.

In Figure 1 (c), the sequence $s = <(19)(23)(22)>$ is contained in s_3 end s_7. If *minsup* is set to 2, s is frequent. Moreover, s is the maximal subsequence shared by s_3 and s_7, s is thus a closed sequential pattern.

Given a sequential data set, there is a unique ordered set that describes the inherent lattice structure defining natural groupings and relationships among the objects and their related sequences [10]. This structure is called a sequence concept lattice. Each element of the lattice is a couple, called sequence formal concept, composed of a set of objects O (the extent) and a set of sequences S (the intent). We call sequential formal concepts, the sequence formal concepts that have at least *minsup* objects in their extent, and sequential concept lattice, the lattice formed using the sequential formal concepts.

In the example (Figure 1 (c), *minsup*=2), the sequential formal concepts are: $(\{s_3, s_7\}; \{<(19)(23)(22)>\})$, $(\{s_3, s_7, s_{10}\}; \{<(19)(23)>\})$, $(\{s_4, s_{10}, s_{12}\}; \{<(14)>\})$, $(\{s_4, s_{12}\}; \{<(14)(10)>\})$, etc.

2.3 Emerging Patterns

Given two data sets $\mathcal{D}_i$, $\mathcal{D}_j$, the quantitative evaluation of the contrast between data sets brought by a pattern is measured by its growth rate [7]. The *growth rate* of a pattern X from $\mathcal{D}_j$ to $\mathcal{D}_i$ is defined by (1).

$$GR_{j,i}(X) = \frac{|\mathcal{D}_j|}{|\mathcal{D}_i|} \times \frac{support(X, \mathcal{D}_i)}{support(X, \mathcal{D}_j)}. \tag{1}$$

The more $GR_{j,i}(X)$ is high, the more X characterizes $\mathcal{D}_i$ compared to $\mathcal{D}_j$. Given a threshold value $mingr > 1$, a pattern X is an *emerging pattern* from $\mathcal{D}_j$ to $\mathcal{D}_i$ if $GR_{j,i}(X) \geq mingr$. An important class of emerging patterns are jumping emerging patterns which correspond to a subset of patterns that is present in one class and absent from the other. If $GR_{j,i}(X) = +\infty$, X is said to be a *jumping emerging pattern* from $\mathcal{D}_j$ to $\mathcal{D}_i$.

Considering sequential patterns and the example of Figure 1 (c), we observe that $s = <(19)(23)(22)>$ is not present in Day2 ($support(s, Day2) = 0$) whereas its support in Day1 is equal to 2. Thus, s is a jumping emerging sequential pattern from Day2 to Day1.

2.4 Predictive Models and Markov Chains

Markov chains are a mathematical formalism from probability theory which is used to model the possible states of a system and the transitions among these states over time [22]. Figure 2 presents a simple transition diagram of a Markov model. The nodes represent the states of the model, and the arrows represent the transitions.

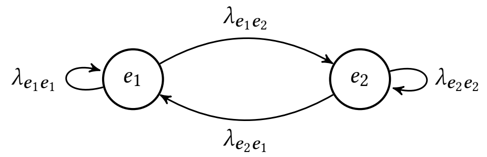


Figure 2: Markov chain example.

The change from the state e_1 to state e_2 is activated by the occurrence of the transition with probability $\lambda_{e_1e_2}$. Let us note that the transition rates are associated in case of continuous time, while we have probabilities in discrete time. The duration of residence in each state are random variables of an exponential distribution. At an instant of time, the change to a next state depends only on the current state and not on the time elapsed in that state. Markov chain models can also be represented by a transition matrix. For example, for the two states e_1, e_2, Four combinations are possible and can be represented in a transition matrix as follows:

$$\mathrm{P} = \begin{array}{cc} & \begin{array}{cc} e_1 & e_2 \end{array} \\ \begin{array}{c} e_1 \\ e_2 \end{array} & \begin{bmatrix} \lambda_{e_1e_1} & \lambda_{e_1e_2} \\ \lambda_{e_2e_1} & \lambda_{e_2e_2} \end{bmatrix} \end{array} \tag{2}$$

The sum of all the probabilities of the transitions from a state must be 1.0. Two types of prediction can be obtained from such models: long term and short term predictions. A stationary analysis for Markov models allows to make a long-term prediction. This is the probability that remains unchanged as time progresses. To compute the stationary probabilities, a vector π which satisfies the equation $\pi.P$=0 is needed. This vector π can be found by two different ways: iterative methods (Gauss-Seidel method, Power method, etc.) or direct methods [22]. While the transient analysis allows to predict the evolution of the model from a given state and for a specified period, mainly it represent the distribution of the system at a given time (short-term prediction). In order to find the transient probabilities, the uniformization method [11] can be employed to compute how many $\pi = P(\pi\text{-}1)$ are necessary to arrive to 1 time unit from the initial state. The transient distribution at time t is obtained by computing an approximation to the infinite summation. Equation 3 presents the uniformization formula where $\pi(0)$ is an initial distribution, and $\frac{(\Gamma t)^k}{k!}e^{-\Gamma t}$ is the probability that a Poisson random variable with uniform rate parameter Γ takes the value k (see Equation 4).

$$\pi(t) = \sum_{k=0}^{\infty} \pi(0)P^k \frac{(\Gamma t)^k}{k!} e^{-\Gamma t} \tag{3}$$

$$\mathbb{P}\{N(t) = k\} = \frac{(\Gamma t)^k}{k!} e^{-\Gamma t} \quad k \geq 0,\ t \geq 0 \tag{4}$$

The proposed method in this paper is concerned with Discrete Time Markov Chain (DTMC) where the state of a system is observed at discrete set of times, including periodicity and recurrence in a finite state-space case.

3 RELATED WORK

Markov chains are often used in a traffic prediction context contrary to sequential patterns. Sequential pattern mining is used in a more general context to analyze spatio-temporal data.

In [24] the authors adapted the classical closed frequent sequence pattern mining algorithm CloSpan to moving trajectory frequent pattern mining. This algorithm, called MTCloSpan, is tested according to the *minsup* value and the characteristics of trajectories (number and length). The authors of [15] proposed an algorithm, called CST-SPMiner, to discover all participation index strong spatio-temporal sequential patterns from event data. Nevertheless, event data are quite different to moving object trajectory data and this work can not be used in our context. Sequential patterns are very usual. However, our method extracts more than sequential patterns, we compute sequential formal concepts which are more informative (transactions IDs, sequence sets) and allow easier exploitation.

Wang et al. proposed a periodic pattern mining of structural evolution in the case of complex trajectories [23]. Mining periodic patterns from spatio-temporal trajectories reveals information about people's regular and recurrent movements and behaviors. In [12], the authors proposed a method to mine spatio-temporal periodic patterns in the traffic data and use these periodic behaviors to summarize the huge road network by clustering. In our work, we do not interest in periodicity but in evolution of extracted patterns over time (between two studied time windows, are they emerging, jumping, lost, etc.?). The evolution types of our method characterize the traffic status trends. We want to predict the traffic status in the next future.

Many recent works has been devoted to develop methods based on Markov models in order to predict, for instance, traffic conditions, congestions, routes and destinations. In [20], the authors proposed a probabilistic approach using Hidden Markov Model (HMM) to model the stochastic variation of traffic conditions and predict the traffic conditions over short time periods. The model defines traffic states based on speed observations of vehicles. Zaki et al. presented in [27] a model, based on HMM, to define the traffic states during peak hours in two dimensional space. The model uses mean speed and contrast to capture the variability in traffic patterns.

Most of works based on vehicles speed use sensors at specific road places. We can say that these methods are local while our approach is global. Our method operates at a more macroscopic level by using any mobility GPS data.

Bouyahia et al. [4] presented a two stage method for predicting traffic congestions. The first stage uses a Markov random field to model and predict the propagation of traffic congestion. The road network is assimilated to a graph. The second stage allows to organize the deployment of the traffic regulation resources using a Markov decision process. In [21], Rathore et al. proposed a framework for trajectory prediction from GPS data. The framework is based on trajectory clustering to find frequent route patterns. For each cluster of trajectories, a Markov chain model is trained to make future trajectory prediction. As our method, a special step is done before training the Markov models. In our case, this is the computation of the sequential formal concepts and the detection of their evolution types.

The previously mentioned works show that Markov models are widely used in a traffic context. Nevertheless, to the best of our knowledge, there is no work combining sequential patterns, detection of patterns evolution, and Markov models to predict trends and traffic evolution.

4 SP4TP METHOD

In this section, we present the proposed method, called SP4TP (Sequential Patterns for Traffic Prediction), which considers extracting traffic evolution status in a study area to make predictions. Figure 3 shows a general overview of this method. The main steps are:

- Spatio-temporal preprocessing (see Section 4.1),
- Pattern mining and status detection (see Section 4.1),
- Region traffic model construction (see Section 4.2),
- Prediction (see Section 4.2).

For a given region and two time windows, the method computes the closed formal concepts (i.e., closed sequential patterns associated to sequence IDs) and detects their evolution types between the times windows. These evolution types (considered as traffic status) are used as input to train the proposed Markov model, which will use to predict the traffic status in the next future windows.

4.1 Pattern Mining and Status Detection

In order to extract specific region traffic status, we applied the method mentioned in Section 2.1. The studied area is divided into grid cells according to a spatial granularity value. Trajectories are then mapped in the discretized area. This discrete representation of trajectory is used to compute the sequential formal concepts (see Section 2.2) according to *minsup* (the minimum threshold value to consider the traffic).

Let (O, S) be a sequential formal concept, θ be the minimum threshold value of emergence, and ϵ be the error tolerance. In this context, the pattern evolution of each concept (O, S) between time i and time i-1 is detected by computing K_i, an indicative value of S, between time i and time i-1 (see Equation 5). K_i is related to the growth rate of a pattern (see Section 2.3).

$$K_i(O,S) = \frac{count(O,t_i)}{count(O,t_{i-1})} \tag{5}$$

Where $count(O, t_j)$ is the number of trajectories of O labeled by the corresponding t_j. We apply the following rules based on the K_i value. If $K_i(O, S) = 0 \pm \epsilon$, then the type of S is *lost.* If $((K_i(O,S) > \theta) \wedge (\theta = 1)) \vee ((K_i(O,S) \geq \theta) \wedge (\theta > 1)))$ then the type of S is *emerging.* If $K_i(O, S) < \theta$, then the type of S is *decreasing.* If $K_i(O, S) = 1 \pm \epsilon$, then the type of S is *latent.* If $K_i(O, S) = +\infty$, then the type of S is *jumping.* In a more intuitive way, *Emerging* means that the presence of the pattern increased in t_{i+1} compared to t_i; *Decreasing* means that the presence of the pattern decreased in t_{i+1} compared to t_i; *Latent* means that the presence of the pattern is quite similar in both time. *Jumping* means the pattern which was absent in t_i , appeared in t_{i+1}. *Lost* means that the pattern disappeared in t_{i+1}.

Table 1 presents some examples of evolution type detection from the sequential data in Figure 1 (c). The sequence $<(14)(10)>$ has been detected as *Latent*, K_2 is equal to 1. This sequence appears in the same way at time 1 and time 2.

Table 1: Some sequential formal concepts and their evolution types obtained from Figure 1 (c) with $minsup = 2$, $\theta = 1$, and $\epsilon = 0$

Extent (O)	Intent (S)	K_2	Evolution type
$\{s_3, s_7\}$	$\{<(19)(23)(22)>\}$	0	Lost
$\{s_3, s_7, s_{10}\}$	$\{<(19)(23)>\}$	0.5	Decreasing
$\{s_4, s_{10}, s_{12}\}$	$\{<(14)>\}$	2	Emerging
$\{s_4, s_{12}\}$	$\{<(14)(10)>\}$	1	Latent

The status of a grid cell is set by considering the evolution type of all the sequential formal concepts that contain this grid cell in its intent and by applying a majority vote. Let ST be the set of status labels: ST=$\{$*Latent* (LA), *Emerging* (E), *Decreasing* (D), *Jumping* (J), *Lost* (LO), *Nothing* ($\emptyset$)$\}$. For each grid cell g_i, we determine the status vector $V_{g_i} =< ST_{t_{min}}, ST_{t_2}, ..., ST_{t_{max}} >$ where ST_{t_j} is the detected status for g_i at time t_j, and $[t_{min}, t_{max}]$ is the studied time interval. This vector expresses a sequence of changing states over a discrete time.

Figure 4 presents an example to determine status vectors. The status vector of g_{13} is $V_{g_i} =< J, E, E, E, ., ., . >$. For t_3, there are three detected status: E, E, and LA. A majority voting process is needed and gives E as result.

Status vectors are used in the next step of our method as input to train the proposed Markov model.

4.2 Modeling Traffic Status by Markov Chains

To predict traffic status, we propose a method based on Markov model representations. Our method uses a specific area and a time window as inputs for studying each grid cell pattern trends. The proposed algorithm that builds the model is composed of three main steps. The first step leads to a graph, where nodes are status of a grid cell and edges represent the transition between two status. The strength of a given transition between nodes A and B is represented by the parameter λ_{ab}. The role of the second step is to compute values of all parameters λ. The last step computes the transits state probability vector for an initial state. More details about these three steps are given below.

Step 1: Considering the set of status ST (see Section 4.1), we construct the structure of the model presented as a graph in Figure 5, which represents the Discrete-Time Markov Chain model for each grid cell. The corresponding transition matrix ($n \times n$ with n=6) is defined in Equation 6. Let us remark that the real-world modeling leads to ignore some transitions. For instance, there is no transition from LO to D because it is not possible to switch from *Lost* status (LO) to *Decreasing* status (D).

$$\mathrm{P} = \begin{array}{c|cccccc} & \mathrm{E} & \mathrm{D} & \mathrm{LA} & \mathrm{J} & \mathrm{LO} & \emptyset \\ \hline \mathrm{E} & \lambda_{ee} & \lambda_{ed} & \lambda_{ela} & 0 & \lambda_{elo} & \lambda_{e\emptyset} \\ \mathrm{D} & \lambda_{de} & \lambda_{dd} & \lambda_{dla} & 0 & \lambda_{dlo} & \lambda_{d\emptyset} \\ \mathrm{LA} & \lambda_{lae} & \lambda_{lad} & \lambda_{lala} & 0 & \lambda_{lalo} & \lambda_{la\emptyset} \\ \mathrm{J} & \lambda_{je} & \lambda_{jd} & \lambda_{jla} & 0 & \lambda_{jlo} & \lambda_{j\emptyset} \\ \mathrm{LO} & \lambda_{loe} & 0 & 0 & \lambda_{loj} & 0 & \lambda_{lo\emptyset} \\ \emptyset & \lambda_{\emptyset e} & 0 & \lambda_{\emptyset la} & \lambda_{\emptyset j} & 0 & \lambda_{\emptyset\emptyset} \end{array} \tag{6}$$

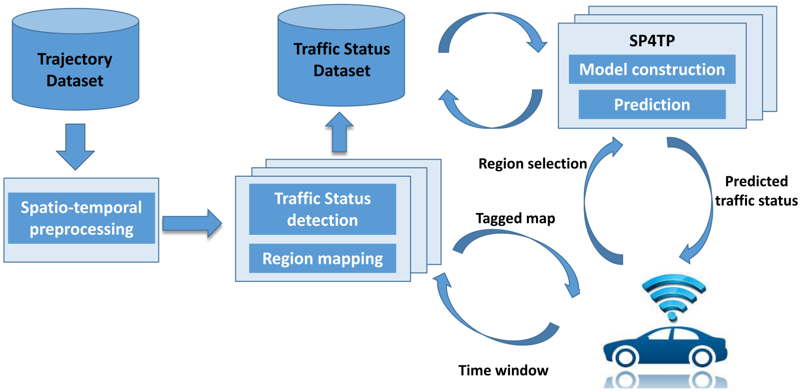
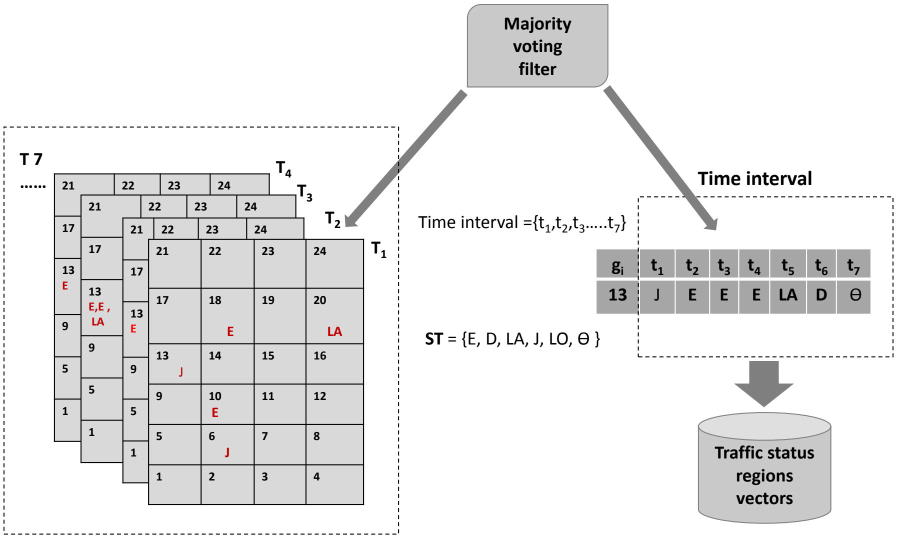


Figure 3: Overview of the proposed method.



g_i	t_1	t_2	t_3	t_4	t_5	t_6	t_7
13	J	E	E	E	LA	D	Θ

Figure 4: Example of grid computed status vector.

We will use two indexes i and j for accessing to a transition value λ between the nodes A and B corresponding to i and j, respectively. For example, p_{12} corresponds to λ_{ed} (the transition from E to D).

Step 2: During this second step, we compute the transition probabilities λ. A discrete-time Markov chain is a process whose state space is finite with time interval indexes as $t = (1, 2, \ldots)$. The generated discrete time vector of status for each grid cell (see Figure 4) has randomness and the probability to have a certain state that depends only on the current state and not the whole previous states in the generated vector. P is the transition probability matrix where each p_{ij} representing the probability of moving from state i to state j. The matrix elements are not negative, and $\sum_{j=1}^{n} p_{ij} = 1\ \forall i$. Thus, the probability p_{ij} is given by Equation 7.

$$p_{ij} = \frac{f_{ij}}{\sum_{k=1}^{n} f_{ik}} \tag{7}$$

Where f_{ij} is the number of occurrences of the transitions from i to j observed in the data, and $\sum_{k=1}^{n} f_{ik}$ is the sum of the frequencies of i to all other states k.

In our context, a transition represents a change in traffic status from one time interval to another. For each transition, a probability is associated which is the ratio of the number of changes from state i to state j divided by the total number of changes from state i.

Step 3: In the last step, we calculate the transits state probability vector for a given initial state s_0 which is defined according to the requested time to be predicted. Our assumption is that the detected traffic status at some point in the future (t+1) can be determined as a function of current traffic status (t), more precisely: $p_{ij} = \mathbb{P}(V_{n+1} = j \mid V_n = i)$, which is one step transition probability.

Given a positive integer m, the m-step transition probability $p_{ij}^{(m)}$ is the probability to reach the state j after m steps starting from the state i, $p_{ij}^{(m)} = \mathbb{P}(V_{n+m} = j \mid V_n = i)$. In this work, we are

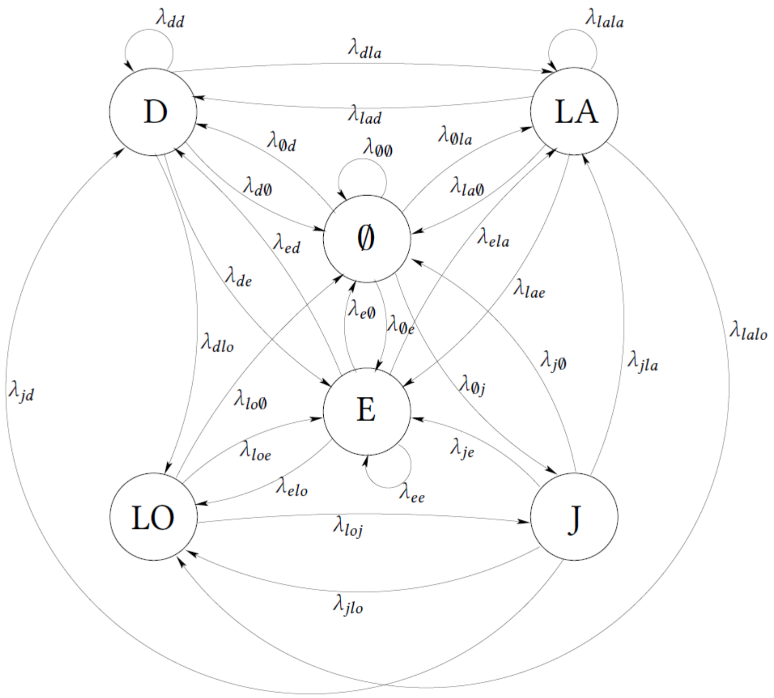


Figure 5: Proposed model for each grid cell.

interested in computing the traffic status after m steps. Thus, the uniformization method (see Equation 3 in Section 2.4) is used to solve the model and obtain the vector of transient probability.

Let us consider the example of the grid cell g_{13} (see Figure 4). The vector $V_{g_{13}} =< J, E, E, E, LA, D, \emptyset >$ contains the observed status for each time t. From this vector, the transition probabilities are computed as described in Step 2. Starting from s_0=J in t_1, the vector π is computed by applying the power method and uniformization method (there are 16 iterations). The maximal value in this vector is obtained for the transition $j \longrightarrow e$ (the value is 0.522977). Thus, the predicted status for the next time t_2 is E (i.e., *Emerging*).

5 EXPERIMENTS

This section presents the experiments performed on two real-world data sets. After the description of the data sets, we describe the experimental protocol and discuss the results.

5.1 Data

In the experiments, we have used two real-world taxi mobility data sets from two cities: Beijing (China) and San Francisco (USA). Table 2 provides a summary of those data sets.

T-Drive data set contains a large amount of taxi GPS trajectories collected in Beijing by Microsoft Research Asia [26]. The original data set contains the trajectories of 10,357 taxis during one week from 2 February 2008 to 8 February 2008. The total number of points is about 15 million and the total distance of the trajectories reaches 9 million kilometers. The GPS coordinates were returned every 5 to 10 seconds. In order to perform our experiments, we have chosen 222 taxis randomly.

Table 2: Taxi mobility data sets

Characteristics	Beijing (T-Drive)	San Francisco
Measurement	GPS	GPS
Number of samples	15 million	11 million
Duration	1 week	24 days
Number of taxis	10,357	500

The second data set is *Taxi cabs San Francisco* which is available from the CRAWDAD website [18]. San Francisco data set includes 500 taxis trajectories over 24 days during the period from 17 May 2008 to 10 June 2008 in the San Francisco Bay Area. This data set contains approximately 11 million taxi GPS samples, with a median time gap between two consecutive GPS measures of 60 seconds. After a preprocessing step of the data, we have obtained 455 active taxis.

5.2 Protocol

The following experimental protocol has been applied to each data set. First, we started by preprocessing all the trajectory data. The trajectories were segmented according to the time granularity value set to 12 hours (AM, PM). The study areas were divided into regions by setting the spatial granularity value to 15, 30, 60, or 120 meters.

After this preprocessing step, we detected the traffic status of each region between each pair of contiguous time windows (see Section 4.1). For this, we set *minsup* to 10 for T-Drive and *minsup* to 45 for San Francisco data set, $\theta = 1$, and $\epsilon = 0$. Let us remark

Table 3: Global average accuracy for T-Drive (Beijing) Data

Spatial gr.	Baseline	SP4TP (it.)	SP4TP (seq.)
15 m	0.747	0.921	**0.962**
30 m	0.785	0.906	**0.942**
60 m	0.807	0.902	**0.929**
120 m	0.831	0.900	**0.920**

Table 4: Global average accuracy for San Francisco Data

Spatial gr.	Baseline	SP4TP (it.)	SP4TP (seq.)
15 m	0.8984	0.922	**0.924**
30 m	0.901	0.909	**0.910**
60 m	**0.899**	0.897	**0.899**
120 m	0.903	0.902	**0.909**

that the parameter setting is proper to each data set and requires some tests. We present here the best obtained results.

Considering the previous detected traffic status as data, the Markov model has been built for each region (see Section 4.2). The proposed method have been applied by changing the start state s_0 during a time interval of 7 days for T-drive, and 24 days in San Francisco data. From t_i, the traffic state of t_j has been predicted, $i = 1 \dots 6$ and $j = 2 \dots 7$ for T-Drive. $i = 1 \dots 23$ and $j = 2 \dots 24$ for San Francisco data.

For the evaluation step, we have compared the predicted status to the real status. We have computed the local accuracy values, the accuracy values for each traffic status and the global average accuracy values. We have compared our proposition to two other approaches to detect traffic status: a baseline approach that does not consider patterns (only singletons), and an itemset approach that does not consider the order of items. In each case, we have used our Markov model and only the way to obtain the traffic status are different. So, we evaluated the gain of using our approach based on sequential patterns.

We have implemented our proposition mainly with JAVA. We have used the SPMF library [8] (developed in JAVA) for computing patterns, and the PEPS software tool [5] (developed in C++) for solving Markov models. All the experiments were performed on an Intel Xeon X5560 2.8GHz with 16GB of memory.

5.3 Results

Tables 3 and 4 present the global average accuracy obtained by applying our proposition (noted SP4TP seq.) and the two other methods: Baseline and Itemsets (noted SP4TP it.) on T-Drive and San Francisco data sets. We can observe that if the spatial granularity value decreases, the accuracy tends to increase for SP4TP (it. and seq.). Nevertheless, the spatial granularity value can not be too low because there will be some difficulties to detect patterns (high number of regions, execution time, memory space). For the Baseline approach, if the spatial granularity value decreases, the accuracy tends to decrease. The absence of relations between grid cells (i.e., patterns) is the explanation. For T-Drive (Beijing), our proposition outperforms the frequent pattern and the baseline approach for all different spatial granularity values. For San Francisco data set, our proposition has obtained best results for low spatial granularity values but the gain is not very significant. We can explain that by the relative quality of the collected data of San Francisco, especially the high time between two GPS measures (this can be a problem to correctly detect patterns).

Table 5: Average accuracy per status for T-Drive (Beijing) Data

Status	Baseline	SP4TP (it.)	SP4TP (seq.)
Emerging	0.653	0.775	**0.849**
Decreasing	0.754	0.936	**0.945**
Latent	0.784	0.969	**0.987**
Lost	0.880	**1**	0.999
Jumping	0.754	**0.987**	0.986
Nothing	**0.929**	0.778	0.863

Table 6: Average accuracy per status for San Francisco Data

Status	Baseline	SP4TP (it.)	SP4TP (seq.)
Emerging	0.732	0.827	**0.836**
Decreasing	0.729	0.790	**0.797**
Latent	0.944	**0.990**	0.988
Lost	0.997	**1**	**1**
Jumping	0.997	**1**	**1**
Nothing	**0.999**	0.837	0.841

Tables 5 and 6 present the average accuracy values per status obtained with T-Drive and San Francisco data set, respectively. As we can see, the *Lost* and *Jumping* status are particular. They are very few and well predicted (almost no errors). We obtained very high values. We can say that, here, these status are not very significant. Let us focus on *Emerging*, *Decreasing* and *Latent*. SP4TP (especially with sequences) outperforms the Baseline method which has its worst results. We can remark that the Baseline method has obtained the best result for the status *Nothin* but this status is not very informative and useful for user applications. For example, Figure 6 presents a tagged maps of Beijing. Regions having the *Nothing* status are not colored because we cannot conclude on them. To analyse the traffic or to establish a travel, a user or an application will naturally lean on the other status (*Emerging*, *Jumping*, ...).

6 CONCLUSION

In this paper, we have proposed a new method for predicting the traffic status in the next future. This method, called SP4TP, leverages closed sequential patterns, emerging patterns and Markov chains. The experimental results on real-world data sets from Beijing and San Francisco have shown that our proposition based on sequential patterns (via sequential formal concepts) outperforms the alternative approaches using itemsets or not using patterns.

In future work, we will perform other experiments with more data sets and compare with other existing systems. Furthermore, we will improve and extend our model to encode the relations between regions (for instance, related to road networks). For this, we can use the SAN (Stochastic Automata Networks) formalism [19] which are able to model the system by the composition of a set of sub-system and it is equivalent to Markov Chains.

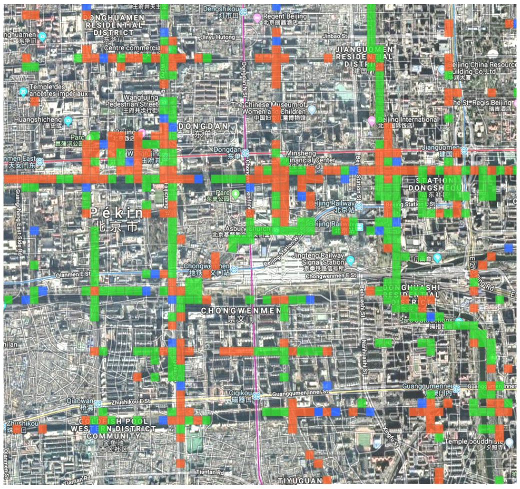


Figure 6: An example of tagged map of Beijing (Blue=Latent, Green=Emerging, Red=Decreasing).

REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami. 1993. Mining association rules between sets of items in large database. In *ACM SIGMOD Int. Conf. on Management of Data*. Washington DC, USA, 207–216.

[2] F. AlMuhisen, N. Durand, and M. Quafafou. 2018. Detecting behavior types of moving object trajectories. *Int. Journal of Data Science and Analytics* 5, 2 (March 2018), 169–187.

[3] F. AlMuhisen, N. Durand, and M. Quafafou. 2018. Sequential Formal Concepts over Time for Trajectory Analysis. In *IEEE/WIC/ACM International Conference on Web Intelligence (WI)*. Santiago, Chile, 598–603.

[4] Z Bouyahia, H Haddad, N Jabeur, and A Yasar. 2019. A two-stage road traffic congestion prediction and resource dispatching toward a self-organizing traffic control system. *Personal and Ubiquitous Computing* 23 (2019), 909–920.

[5] L. Brenner, P. Fernandes, B. Plateau, and I. Sbeity. 2007. PEPS 2007 - Stochastic Automata Networks Software Tool. In *4th Int. Conf. on the Quantitative Evaluation of SysTems (QEST)*. Edimbourgh, UK, 163–164.

[6] E. Cesario, C. Comito, and D. Talia. 2014. *Trajectory data analysis over a cloud-based framework for smart city analytics*. Springer, 143–162.

[7] G. Dong and J. Li. 1999. Efficient mining of emerging patterns: Discovering trends and differences. In *5th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*. San Diego, CA, USA, 43–52.

[8] P. Fournier-Viger, C.W. Lin, A. Gomariz, T. Gueniche, A. Soltani, Z. Deng, and H. T. Lam. 2016. The SPMF Open-Source Data Mining Library Version 2. In *19th European Conference on Principles of Data Mining and Knowledge Discovery*. Riva del Garda, Italy, 36–40.

[9] P. Fournier-Viger, J. C.-W. Lin, R. U. Kiran, Y. S. Koh, and R. Thomas. 2017. A survey of sequential pattern mining. *Data Science and Pattern Recognition* 1, 1 (2017), 54–77.

[10] G. C. Garriga. 2013. Lattice theory for sequences. In *Formal Methods for Mining Structured Objects*. Springer, 21–37.

[11] A. Jensen. 1953. Markoff chains as an aid in the study of Markoff processes. *Scandinavian Actuarial Journal* (1953), 87–91.

[12] T. Jindal, P. Giridhar, L.-A. Tang, J. Li, and J. Han. 2003. Spatiotemporal Periodical Pattern Mining in Traffic Data. In *2nd ACM SIGKDD International Workshop on Urban Computing*. Chicago, Illinois, USA, 1–8.

[13] Y. Lassoued, J. Monteil, Y. Gu, G. Russo, R. Shorten, and M. Mevissen. 2017. A Hidden Markov Model for Route and Destination Prediction. In *Int. Conf. on Intelligent Transportation Systems (ITSC)*. Yokohama, Japan.

[14] M. Lin and W.-J. Hsu. 2014. Mining GPS data for mobility patterns: A survey. *Pervasive and Mobile Computing* 12 (2014), 1–16.

[15] P. S. Maciag, M. Kryszkiewicz, and R. Bembenik. 2019. Discovery of closed spatio-temporal sequential patterns from event data. *Procedia Computer Science* 159 (2019), 707–716.

[16] A. M. Nagy and V. Simon. 2018. Survey on traffic prediction in smart cities. *Pervasive and Mobile Computing* 50 (2018), 148–163.

[17] G. Pan, G. Qi, W. Zhang, S. Li, Z. Wu, and L. T. Yang. 2013. Trace analysis and mining for smart cities: Issues, methods, and applications. *IEEE Communications Magazine* 51, 6 (June 2013), 120–126.

[18] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser. 2009. CRAWDAD dataset epfl/mobility (v. 2009-02-24). https://crawdad.org/epfl/mobility/20090224/cab. traceset: cab.

[19] B. Plateau and W. J. Stewart. 2000. *Stochastic Automata Networks*. Springer, 113–151.

[20] Y. Qi and S. Ishak. 2014. A Hidden Markov Model for short term prediction of traffic conditions on freeways. *Transportation Research Part C* 43 (2014), 95–111.

[21] P. Rathore, D. Kumar, S. Rajasegarar, M. Palaniswami, and J. C. Bezdek. 2019. A Scalable Framework for Trajectory Prediction. *IEEE Transactions on Intelligent Transport Systems* 20 (2019), 1–15.

[22] W. J. Stewart. 1994. *Introduction to the numerical solution of Markov chains*. Princeton University Press.

[23] H. Wang, Y. Du, J. Yi, N. Wang, and F. Liang. 2020. Mining Evolution Patterns from Complex Trajectory Structures. *ISPRS Int. Journal of Geo-Information* 9, 7 (2020).

[24] L. Wang, K. Hu, T. Ku, and J. Wu. 2012. Discovering closed frequent patterns in moving trajectory database. In *14th International Conference on Communication Technology*. Chengdu, China, 567–572.

[25] H. Yuan and G. Li. 2021. A Survey of Traffic Prediction: from Spatio-Temporal Data to Intelligent Transportation. *Data Sci. Eng.* 6 (2021), 63–85.

[26] N. J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. 2010. T-drive: Driving directions based on taxi trajectories. In *Int. Conf. on Advances in Geographic Information Systems*. San Jose, California, USA, 99–108.

[27] J. F. Zaki, A. M. Ali-Eldin, S. E. Hussein, S. F. Saraya, and F. F. Areed. 2020. Traffic congestion prediction based on Hidden Markov Models and contrast measure. *Ain Shams Engineering Journal* 11 (2020), 535–551.

[28] X. Zhang, S. Wang, and Y. Tian. 2014. A research on driving condition prediction for HEVs based on Markov chain. *Automotive Engineering* 36, 10 (2014), 1216–1220.

[29] S.-X. Zhao, H.-W. Wu, and C.-R. Liu. 2019. Traffic flow prediction based on optimized hidden Markov model. *Journal of Physics Conference Series* 1168, 5 (2019), 052001.

[30] Y. Zheng. 2015. Trajectory data mining: An overview. *ACM Trans. Intell. Syst. Technol.* 6, 3 (2015), 29–41.