

RESEARCH

Open Access



Scalable prediction by partial match (PPM) and its application to route prediction

Vishnu Shankar Tiwari*, Arti Arya and Sudha Chaturvedi

*Correspondence:
vishnustiwari@gmail.com
Visvesvaraya Technological
University, Bangalore,
Karnataka, India

Abstract

Route prediction plays a vital role in many important location-based applications such as resource prediction in grid computing, traffic congestion estimation, vehicular ad hoc networks, and travel recommendation. The goal of this work is to design a scalable route prediction application based on prediction by partial match (PPM) modeling of user travel data. PPM is one of the widely used techniques for text compression as well as string sequence indexing and for prediction. PPM tree construction from the huge volume of data by sequential processing is time consuming in practical implementation. Existing techniques are designed for single machine and their implementation on the distributed environment is still a challenge. This work focuses on achieving a horizontal scalability of PPM and addresses various challenges in distributed construction, such as reducing I/O and parallel computation of sequences, and comes up with a final PPM tree in distributed environment without sacrificing accuracy. A huge corpus of GPS data set is map matched to the road network extracted from the OpenStreetMap and the PPM tree is built on the edges of the road network. A two-step construction of the PPM tree is proposed, which is extended to execute over the MapReduce framework. The MapReduce framework running over the Hadoop distributed file system is used for distributed processing. A horizontally scalable PPM model is built and evaluated for route prediction from a huge corpus of historical GPS traces. Data sets used are GPS traces and road networks. Both of these used in this work are taken from an openly available corpus. Distributed construction of PPM was proposed and evaluated on Hadoop cluster using MapReduce and the detailed results are presented.

Keywords: PPM, Big data, Scalability, MapReduce, Route prediction

Introduction

Route prediction is a key requirement in many location-based important applications such as vehicular ad hoc networks, traffic congestion estimation, resource prediction in grid computing, vehicular turn prediction, travel pattern similarity, and pattern mining. Route prediction is a problem which deals with, given a sequence of road network graph edges already traveled by the user, predicting the most probable edge of the network to be traveled. Our approach is to build a prediction by partial match (PPM) model from a huge corpus of sequential trajectories traveled by the user in the past. PPM is widely used in various applications in the area of data compression and machine learning (Begleiter et al. 2004). Time-stamped GPS traces are collected over a long period

of time. The chronological huge sequence of GPS traces is broken down into smaller units called trip (Froehlich and Krumm 2008; Tiwari et al. 2013). Trips are mapped to road network graph using map matching process which identifies the object's location on the road network graph (Tiwari et al. 2014; Bernstein and Kornhauser 1996; Zhou and Golledge 2006). PPM tree-based model is constructed from trips composed of an ordered sequence of road network edges. Given a trajectory traveled by the user, a lookup is done in the PPM tree-based model and the most likely edge is found.

Cleary and Witten invented PPM back in (1984). Many versions of PPM evolved thereafter (Moffat 1990; Cleary et al. 1995; Teahan 1995; Schürmann and Grassberger 1996). PPM models learn from historical occurrences of sequences to predict the probability of a specific data appearing after a given data sequence. For experiments in this work, a version PPM-C is used. We explain the process of construction of PPM-C, followed by distributed construction of the same. Real applications using PPM deals with processing of huge data sets, and processing such volume sequentially and coming up with a PPM model is a bottleneck. Attempts have been made to achieve scalability by adding processors and memory (Gilchrist 2004; Joel and Sirota 2012; Effros 2000). However, distributed construction of PPM is still a challenge. In the proposed work, scalability is achieved by decomposing GPS traces into trips and processing them in parallel and finally consolidating them to form the PPM model. A set of user trips is decomposed into smaller sets and ported to compute a module known as mappers. Mappers compute the variable order contexts as key-value pairs. In each case, the key is the context and value is the occurrence frequency in the training set. Key-value pairs from various mappers are emitted to the reducer node. Reducer consolidates the occurrences of various contexts and inserts in the PPM trie. The final tree produced by the reducer is the PPM model which is used for route prediction. The major contribution of this work is a technique of distributed computation of PPM and its application in route prediction. All experiments and implementations are done on real data sets available openly in the public domain.

PPM tree-related work and literature

Prediction by partial match (PPM) is a context modeling-based adaptive statistical data compression technique. It has evolved as a better alternative for solving many problems in the field of biomedical engineering, natural language processing and artificial intelligence. PPM models use a set of historical occurrences of sequences to predict the probability of a specific symbol appearing at a given position in an input stream (Begleiter et al. 2004). Arithmetic encoding was proposed in 1976, after which soon PPM variants PPM-A and PPM-B were invented by Cleary and Witten (1984). These were further improved by Moffat (1990), resulting in PPM-C and PPM-D. PPM-D was proved show a bit more improvement in some cases. All these variants of PPM are more or less same, but only differ in the way probability is computed. In all the cases, the PPM model is a mix of lower-order models. If unsuitable results are found with the higher-order model, then it falls back to lower-order models (context of lesser length). Hiroyuki et al. (2005) presented an unbounded version of PPM known as PPM*, used for the classification of text. However, it proposed the use of finite deterministic contexts. Cleary et al. (1995) also proposed an unbounded variant of PPM, but scalability and parallelism were not

addressed. It was well established that it performs well in compression, language identification, text prediction, word segmentation, text categorization, etc. Gilchrist et al. (2004) proposed parallel computation with more focus on BZIP2 in a multi-processor system. Effros et al. (2004) presented an improvement on PPM, but parallelism was not addressed. Begleiter et al. (2004) further explored PPM and successfully applied it to artificial intelligence (AI) applications including text prediction and music recognition and it worked well. Celikel et al. (2005) applied PPM for language recognition and proved promising results. The objectives of almost all researches on PPM were either improving its accuracy and execution on a single machine or its application in different fields of study. In spite of their huge applicability, parallel execution and PPM model construction were hardly explored. The objective of this research is to come up with a technique for distributed parallel construction of the PPM model tree. The major milestones in PPM are as listed in Table 1.

PPM tree basics

Time-stamped GPS traces are collected over a long period of time. GPS traces are in the form $(x_{t^0}, y_{t^0}, t^0), (x_{t^1}, y_{t^1}, t^1) \dots (x_{t^n}, y_{t^n}, t^n)$, which represents the object's location (x_{t^k}, y_{t^k}) at time t^k . Chronological huge sequences of GPS traces are broken down into smaller units called trips (Froehlich and Krumm 2008; Tiwari et al. 2013). A user trip $T = (p_s, t_s, p_e, t_e)$ is an ordered sequence of GPS location data points $(p_i, t_i) \forall 1 \leq i \leq n$ where p_s, p_e are the start and end positions and t_s, t_e are the start and end time of trips, respectively.

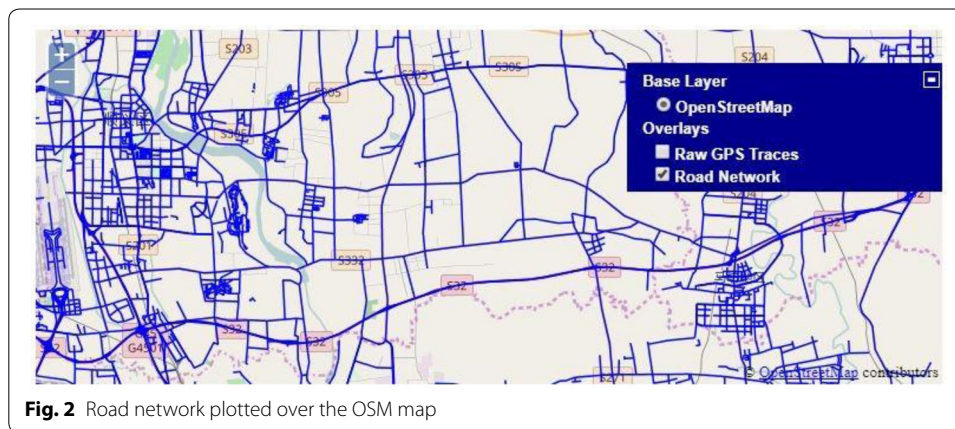
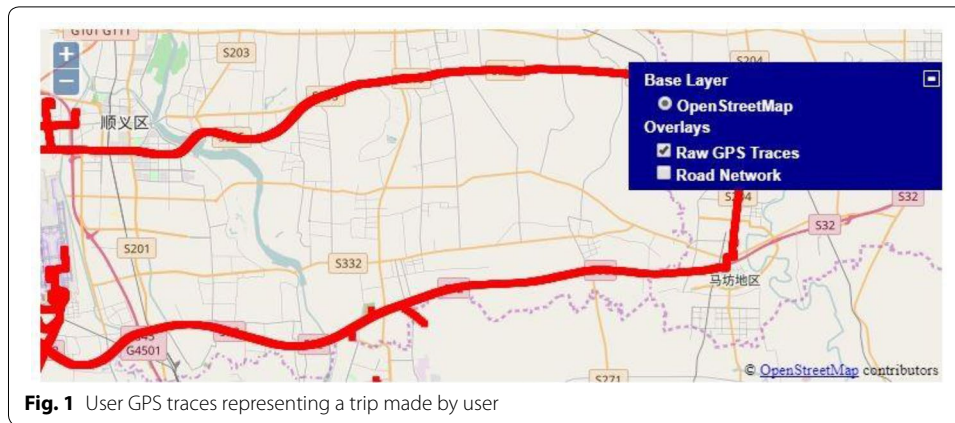
$$T = (x_{t^0}, y_{t^0}, t^0), (x_{t^1}, y_{t^1}, t^1) \dots (x_{t^m}, y_{t^m}, t^m),$$

$$p_s = (x_{t^0}, y_{t^0}), t_s = t^0, \quad p_e = (x_{t^m}, y_{t^m}), t_e = t^m.$$

Two trips T_1 and T_2 are said to be consecutive if the end of the first trip is in the same position as the end of the second trip and there is a time gap between the two. A user trip plotted on OpenStreetMap (OSM) base images is as shown in Fig. 1.

Table 1 Important PPM construction algorithms

	Complexity	Parallel	Probabilistic
Bernstein and Kornhauser (1996)	$O(n^2)$	No	Yes
PPM-A (Cleary and Witten 1984)	$O(n^2)$	No	Yes
PPM-B (Cleary and Witten 1984)	$O(n^2)$	No	Yes
PPM-C (Moffat 1990)	$O(n^2)$	No	Yes
PPM-D (Moffat 1990)	$O(n^2)$	No	Yes
Cleary and Witten (1995)	$O(n^2)$	No	Yes
Teahan (1995)	$O(n^2)$	No	Yes
Joel and Sirota (2012)	$O(n^2)$	No	Yes
Effros (2000)	$O(n^2)$	No	Yes
PPM* (Hiroyuki et al. 2005)	$O(n^2)$	No	Yes
Celikel (2005)	$O(n^2)$	No	Yes
Proposed PPM	$O(n^2)$	Yes	Yes



Trips are mapped to road network graph using map matching process which identifies the object's location on the road network graph (Quddus 2006; Quddus et al. 2006; Greenfeld 2002). An example of a road network extracted from OSM is shown in Fig. 2. Map matching is function f , for which the input is the GPS location and the road network graph, and the output is the edge of the road network.

$$f\left(\left(x_{t^0}, y_{t^0}, t^0\right), \left(x_{t^1}, y_{t^1}, t^1\right) \ldots \left(x_{t^n}, y_{t^n}, t^n\right)\right) \rightarrow S,$$

where sequence S is an ordered sequence of road network edges. Figure 3 shows the GPS traces corrected and mapped to the road network. Let $\Sigma = \{e_1, e_2, e_3, e_4, e_5\}$ be a finite set of all the edges of digitized road network and Σ^* represent all finite length trips possible. Any trip a user makes essentially belongs to Σ^* . Let $X = e_0, e_1, \dots, e_{n-1}$ with $x_i \in \Sigma$ and $X \in \Sigma^*$ be a trip, then the length of the trip is given by $|X| = |e_0, e_1, \dots, e_{n-1}|$.

The ordered arrangement of all sequences $s\sigma$, where σ is the symbol and s is the context of σ , is compact and TRIE is known as the PPM tree. For demonstration purpose, let us assume alphabet set $\Sigma = \{e_1, e_2, e_3, e_4, e_5\}$ and a string $X = e_1, e_2, e_5, e_1, e_3, e_1, e_4, e_1, e_2, e_5, e_1$. All the contexts with length $d=2$ are as shown in Table 2 and the resulting PPM tree is as shown in Fig. 4.

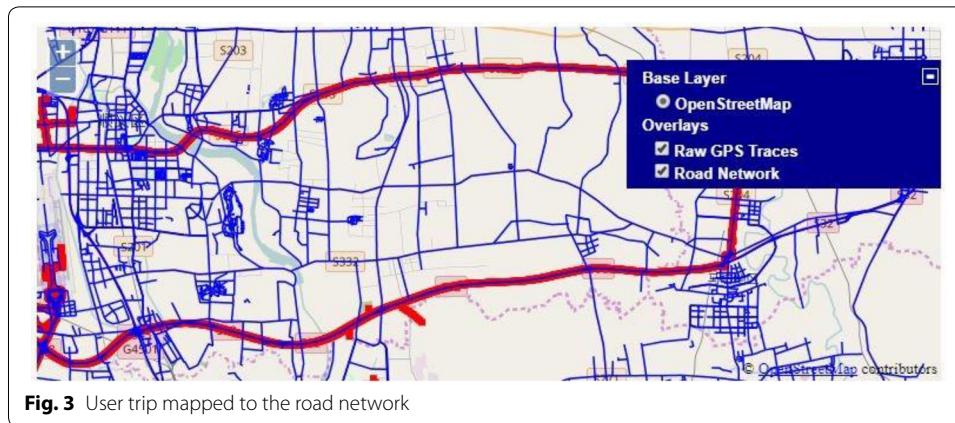
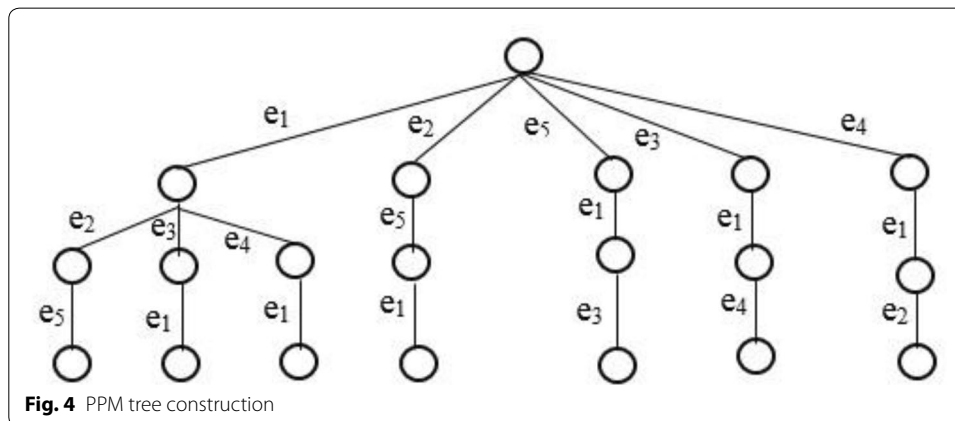


Table 2 All contexts computed for PPM construction

S. no.	D	Context (s)	Symbol (σ)	$s\sigma$
1	2	e_1, e_2	e_5	e_1, e_2, e_5
2	2	e_2, e_5	e_1	e_2, e_5, e_1
3	2	e_5, e_1	e_3	e_5, e_1, e_3
4	2	e_1, e_3	e_1	e_1, e_3, e_1
5	2	e_3, e_1	e_4	e_3, e_1, e_4
6	2	e_1, e_4	e_1	e_1, e_4, e_1
7	2	e_4, e_1	e_2	e_4, e_1, e_2



Prediction by partial match (ppm) tree construction

Two-phase PPM tree construction

We propose a two-step process to compute the PPM trie from user trips. The first phase computes all sequences $s\sigma$, where σ is the symbol and s is the context of σ , and the second phase constructs the trie from sequences computed in first phase. The algorithm scans one alphabet at a time and adds to the dictionary, new phrases which

are shortest and not yet discovered. The subsequence generation process is represented in Algorithm 1.

Algorithm I: Context and frequency calculation

Input: String X and context length $d = \text{constant}$

Output: Data map M' of all contexts including target symbol σ with frequency M' (context, count)

Algorithm:

1. Instantiate an empty map M'
2. Scan over X from left to right and for each substring S' of length $|S'| = d + 1$:
 - I. if $S' \notin M'$ then insert S' as key and 1 as value into map M'
 - II. if $S' \in M'$ then increment value by 1 for key $M'(S')$

The following sequence is used for demonstration: $X = e_1, e_2, e_5, e_1, e_3, e_1, e_4, e_1, e_2, e_5, e_1$. All context s of length $d = 2$ along with target symbol σ denoted by $s\sigma$ computed by Algorithm 1, is as shown in Table 3. The length of string X is denoted by n . All contexts of length d in X can be calculated in linear time $\Theta(n)$ by scanning X from left to right and maintaining a window of size d . The Window is advanced by one unit on scanning one symbol. The maximum number of context strings each of length d that can appear in map is $\Theta(n - d) \approx \Theta(n)$, where $d \ll n$. This can happen only if contexts do not overlap; otherwise in practice, the number of contexts is $\leq \Theta(n)$.

The second phase starts with a tree from scratch and keeps on inserting context sequences $s\sigma$, obtained as input from the first step. For a new context which is not seen earlier, a completely new branch is created. Otherwise a path in tree is searched which is matching/overlapping with current context. All the nodes in overlapping path is increased by frequency of occurrence and remaining nodes are inserted at the end of overlapping path. The process is explained in Algorithm 2. The resultant PPM tree is constructed by Algorithm 2 from the map of context strings in Table 3 including the frequency count.

Algorithm II: PPM Tree Construction

Input: Map M' of all contexts including target symbol σ with frequency $f(\text{context}, \text{count})$

Output: PPM Tree with frequency

Algorithm:

1. Instantiate an empty tree T
2. for each $\langle \text{key}, \text{value} \rangle$ pair $\langle \text{context}_i, \text{count}_i \rangle \in M'(\text{context}, \text{count})$
 - I. Traverse tree T till any branch has overlap with context_i and increment each node by count_i
 - II. For remaining alphabets of context_i fork a branch starting last node till where overlap was found in step I.
 - III. For each node in new branch set count equal to count_i
3. Return resultant tree T

Table 3 All contexts computed by Algorithm 1

S. no.	d	Context (s)	Symbol (σ)	$s\sigma$	Frequency (f)
1	2	e_1, e_2	e_5	e_1, e_2, e_5	2
2	2	e_2, e_5	e_1	e_2, e_5, e_1	2
3	2	e_5, e_1	e_3	e_5, e_1, e_3	1
4	2	e_1, e_3	e_1	e_1, e_3, e_1	1
5	2	e_3, e_1	e_4	e_3, e_1, e_4	1
6	2	e_1, e_4	e_1	e_1, e_4, e_1	1
7	2	e_4, e_1	e_2	e_4, e_1, e_2	1

Table 4 All contexts with frequency computed by m_1

S. no.	d	Context (s)	Symbol (σ)	$s\sigma$	Frequency (f)	$\langle K, V \rangle$
1	2	e_1, e_2	e_5	e_1, e_2, e_5	2	$\langle e_1, e_2, e_5, 2 \rangle$
2	2	e_2, e_5	e_1	e_2, e_5, e_1	2	$\langle e_2, e_5, e_1, 2 \rangle$
3	2	e_5, e_1	e_3	e_5, e_1, e_3	1	$\langle e_5, e_1, e_3, 1 \rangle$
4	2	e_1, e_3	e_1	e_1, e_3, e_1	1	$\langle e_1, e_3, e_1, 1 \rangle$
5	2	e_3, e_1	e_4	e_3, e_1, e_4	1	$\langle e_3, e_1, e_4, 1 \rangle$
6	2	e_1, e_4	e_1	e_1, e_4, e_1	1	$\langle e_1, e_4, e_1, 1 \rangle$
7	2	e_4, e_1	e_2	e_4, e_1, e_2	1	$\langle e_4, e_1, e_2, 1 \rangle$

Table 5 All contexts with frequency computed by m_2

S. no.	d	Context (s)	Symbol (σ)	$s\sigma$	Frequency (f)	$\langle K, V \rangle$
1	2	e_5, e_1	e_3	e_5, e_1, e_3	2	$\langle e_5, e_1, e_3, 2 \rangle$
2	2	e_1, e_3	e_1	e_1, e_3, e_1	2	$\langle e_1, e_3, e_1, 2 \rangle$
3	2	e_3, e_1	e_4	e_3, e_1, e_4	1	$\langle e_3, e_1, e_4, 1 \rangle$
4	2	e_1, e_4	e_1	e_1, e_4, e_1	1	$\langle e_1, e_4, e_1, 1 \rangle$
5	2	e_4, e_1	e_2	e_4, e_1, e_2	1	$\langle e_4, e_1, e_2, 1 \rangle$
6	2	e_1, e_2	e_5	e_1, e_2, e_5	1	$\langle e_1, e_2, e_5, 1 \rangle$
7	2	e_2, e_5	e_1	e_2, e_5, e_1	1	$\langle e_2, e_5, e_1, 1 \rangle$

Distributed construction of the PPM tree

To achieve a distributed construction of the CTW tree-based model, the two-step process described in the earlier section is extended to be executed over the Hadoop cluster leveraging the MapReduce computation framework. The first phase is executed by the mapper module. GPS traces are decomposed into smaller units called trips and map matched to road network. Trips as ordered sequence of road network edges are grouped into smaller sets and processed by mapper module. All the contexts, $s\sigma$, are generated by the mapper for each symbol σ in the trip and are put into a map which stores sequence as key and frequency as value. Implementation of the mapper module is as described in Algorithm 3.

Algorithm III - Mapper function: Context and frequency calculation of local partition

Input: Split chunk S' of original data string S and context length $d = \text{constant}$

Output: Data map M' of all contexts including target symbol σ with frequency $M'(\text{context}, \text{count})$

Algorithm:

1. Instantiate an empty map M'
2. Scan over X from left to right and for each substring S' of length $|S'| = d + 1$:
 - I. if $S' \notin M'$ then insert S' as key and 1 as value into map M'
 - II. if $S' \in M'$ then increment value by 1 for key $M'(S')$
3. Send data map M' to reducer function

To demonstrate the distributed construction of the PPM tree, we take a string below. This will be used as a running example throughout further discussions.

$e_1, e_2, e_5, e_1, e_3, e_1, e_4, e_1, e_2, e_5, e_1, e_3, e_1, e_4, e_1, e_2, e_5, e_1, e_3, e_1.$

For the sake of simplicity and to demonstrate the concept, the input string is split into two chunks. For each of the split, a mapper is instantiated.

Split $S_1 = e_1, e_2, e_5, e_1, e_3, e_1, e_4, e_1, e_2, e_5, e_1$ processed by mapper m_1 ,

Split $S_2 = e_5, e_1, e_3, e_1, e_4, e_1, e_2, e_5, e_1, e_3, e_1$ processed by mapper m_2 .

The output of both mappers m_1 and m_2 are summarized in Tables 4 and 5, respectively. In this example, context σ serves as the key and frequency (f) as value.

The output of the mapper modules is a set of key–value pairs, where the key is the context and value as the frequency is emitted as input to the reducer. The framework does a consolidation by adding the frequencies for each context as key. For example, if from one mapper the value received is $\langle e_1, e_2 | 4 \rangle$ and $\langle e_1, e_2 | 10 \rangle$, then after merging the final entry becomes $\langle e_1, e_2 | 14 \rangle$. It is ensured that each key–value pair is unique during this step. If multiple entries exist for the same key, then consolidation is done before sending it to the reducer. If data do not fit into memory, then it is periodically written to disk (Chang et al. 2008; Jeffrey and Sanjay 2004; Lammel 2008). The reducer starts with a tree from scratch and keeps on inserting context sequences iteratively. For a new context which is not seen earlier, a completely new branch is created. Otherwise a path in tree is searched which is matching/overlapping with current context. All the nodes in overlapping path is increased by frequency of occurrence and remaining nodes are inserted at the end of overlapping path. The result of the consolidation of the output of mappers in Tables 4 and 5 is as shown in Table 6. Implementation of the reducer is as described in Algorithm 4.

Algorithm IV - Reducer function: PPM Tree Construction

Input: Map M' of all contexts including target symbol σ with frequency $f(\text{context}, \text{count})$

Output: PPM Tree with frequency

Algorithm:

1. Instantiate an empty tree T
2. for each $\langle \text{key}, \text{value} \rangle$ pair $\langle \text{context}_i, \text{count}_i \rangle \in M'(\text{context}, \text{count})$
 - i. Traverse tree T till any branch has overlap with context_i and increment each node by count_i
 - ii. For remaining alphabets of context_i fork a branch starting last node till where overlap was found in step I.
 - iii. For each node in new branch set count equal to count_i
3. Return resultant tree T

Route prediction using the PPM tree

The objective is to predict the next edge $\sigma \in E$ on the road network given the user traveled trajectory $S = (x_{t^0}, y_{t^0}, t^0), (x_{t^1}, y_{t^1}, t^1) \dots (x_{t^n}, y_{t^n}, t^n)$, based on information learned from historical user travel data. To predict next edge σ , S is map matched

Table 6 Result of merging of intermediate key/value pairs by MapReduce framework

S. no.	d	Context (s)	Symbol (σ)	Key (k)	Frequencies	$\langle K, \langle \text{sum}(\text{occurrence}) \rangle \rangle$
1	2	e_1, e_2	e_5	e_1, e_2, e_5	2, 1	$\langle e_1, e_2, e_5, \langle 3 \rangle \rangle$
2	2	e_2, e_5	e_1	e_2, e_5, e_1	2, 1	$\langle e_2, e_5, e_1, \langle 3 \rangle \rangle$
3	2	e_5, e_1	e_3	e_5, e_1, e_3	1, 2	$\langle e_5, e_1, e_3, \langle 3 \rangle \rangle$
4	2	e_1, e_3	e_1	e_1, e_3, e_1	1, 2	$\langle e_1, e_3, e_1, \langle 2 \rangle \rangle$
5	2	e_3, e_1	e_4	e_3, e_1, e_4	1, 1	$\langle e_3, e_1, e_4, \langle 2 \rangle \rangle$
6	2	e_1, e_4	e_1	e_1, e_4, e_1	1, 1	$\langle e_1, e_4, e_1, \langle 2 \rangle \rangle$
7	2	e_4, e_1	e_2	e_4, e_1, e_2	1, 1	$\langle e_4, e_1, e_2, \langle 2 \rangle \rangle$

to digitize to road network using the map matching process f as described in earlier sections.

$$f(x_{t^0}, y_{t^0}, t^0), (x_{t^1}, y_{t^1}, t^1) \dots (x_{t^n}, y_{t^n}, t^n) \rightarrow e_i e_{i+1} \dots e_{i+n}.$$

Trajectory S is the converted form of an ordered sequence of road network edges and can be considered as a Markov chain, where the highest possibility of occurrence among all other possibilities is

$$p(\sigma | e_i e_{i+1} \dots e_{i+n}).$$

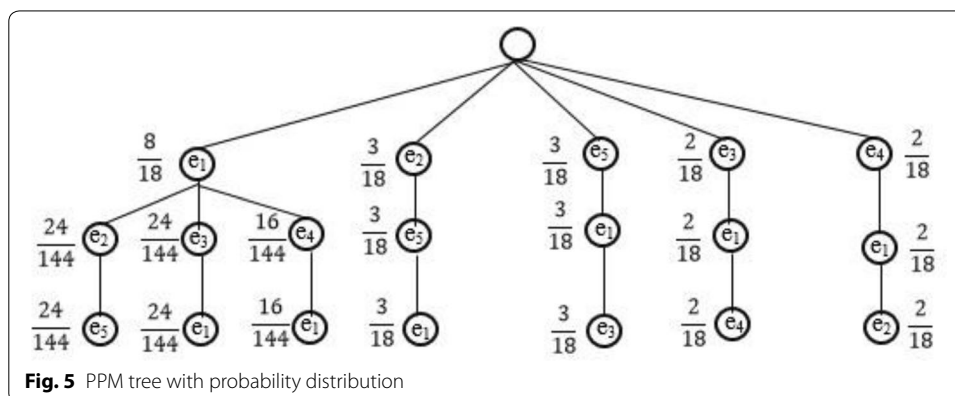
p is the conditional probability of occurrence of σ given the event $e_i e_{i+1} \dots e_{i+n}$ has already occurred. PPM trie constructed has information learned from historical travel data of user. Since PPM is an unbounded Markov model, the corresponding tree may be balanced and each path from the root may be of different length. This makes PPM a variable order Markov model. In the worst case, one has to traverse the longest branch of the PPM tree. If the length of the longest branch of tree is k , then the complexity of the prediction using the PPM trie is $O(k)$. The probabilities of occurrence of each node starting with the root node is as shown in Fig. 5. Route prediction function denoted by a function *Route_Predict* can be represented as:

$$Route_Predict(e_i e_{i+1} \dots e_{i+n}) \rightarrow \sigma.$$

The below cases demonstrate the prediction *Route_Predict* function over the PPM model constructed by Algorithm 4.

Case I: This is the case when the user is at root node which signifies the user has not started travel. We represent the user trajectory by $S = \varepsilon$. From the PPM trie, it can be seen that the various possibilities for traversals are $\{e_1, e_2, e_3, e_4, e_5\}$. The probability for each case is as follows:

$$\begin{aligned} p(e_1|\varepsilon) &= \frac{8}{18}, & p(e_2|\varepsilon) &= \frac{3}{18}, & p(e_3|\varepsilon) &= \frac{2}{18}, \\ p(e_4|\varepsilon) &= \frac{2}{18}, & p(e_5|\varepsilon) &= \frac{3}{18}. \end{aligned}$$



Hence, $Route_Predict(\varepsilon) \rightarrow e_1$.

- Case II: Another case we explore is when edge e_2 has been traversed so far, $S = e_2$. The length of the input trajectory is 1 unit only and consists of a single edge. The candidate edge after e_2 already traversed is only one and is e_5 . In this case, the probability of occurrence of e_5 after e_2 as context is $p(e_5|e_2) = 1$. Hence, $Route_Predict(e_2) \rightarrow e_5$.
- Case III: The next case is when the input trajectory is $S = \{e_1\}$ and only one edge e_1 has been traversed so far. However, there are multiple candidates ($\{e_2, e_3\}$) with high probability after edge e_1 is already traversed. The probabilities of each candidate is as follows:

$$p(e_2|e_1) = \frac{3}{8}, \quad p(e_3|e_1) = \frac{3}{8}$$

Hence, two edges are likely and will be resolved once more edges are traveled.

- Case IV: Next, we consider a case when multiple edges are traveled and the input to $Route_Predict$ function is $\{e_1, e_2\}$. The possible candidate for travel next is edge e_5 , having the said event of traveling over $\{e_1, e_2\}$ already occurred. $p(e_5|e_1, e_2) = 1$ and hence $Route_Predict(e_1, e_2) \rightarrow e_5$.
- Case V: Next, we consider a case when the user has traveled a path which has not yet been seen by the PPM model. For example, if the user has traveled path $\{e_3, e_4\}$ but in the trie no such path exists, this means something which has not occurred in the past. Hence, the prediction function result is $Route_Predict(e_3, e_4) \rightarrow \varepsilon$. This can happen when the user has reached the destination and there is nothing to predict, and in another case it is a new route. In the latter case, new routes when found should be sent to the model for learning.
- Case VI: All the above cases focused on predicting one hop next edge. The same model can be used to predict an end to the end path as well. The input trajectory is ε . The next edge selected is e_1 . From e_2 , the next probable edge is e_5 and so on

Implementation and evaluation

Map data: spatial road network data

OpenStreetMaps provides various kinds of geographical spatial and non-spatial data sets such as water bodies, international boundaries, state boundaries, and road networks. In this work, we use digitized road network data downloaded from OSM. OSM provides open source data under public open content license (<https://www.openstreetmap.org>). Data can be downloaded in a variety of format images, XML files, shape files, etc. We used only road network data from OSM. Data can be downloaded using the OSM interface (<https://www.openstreetmap.org>) if the area is smaller. If the area is larger, for example, the official vendor portal can be used (<https://www.cloudmade.org>). Data are available in various standard formats such as image (.jpg, .png, etc.) or XML which comes with extension.osm. We used.osm format which we parsed using open source tool called Osm2pgsql (wiki.openstreetmap.org/wiki/Osm2pgsql). It is used to

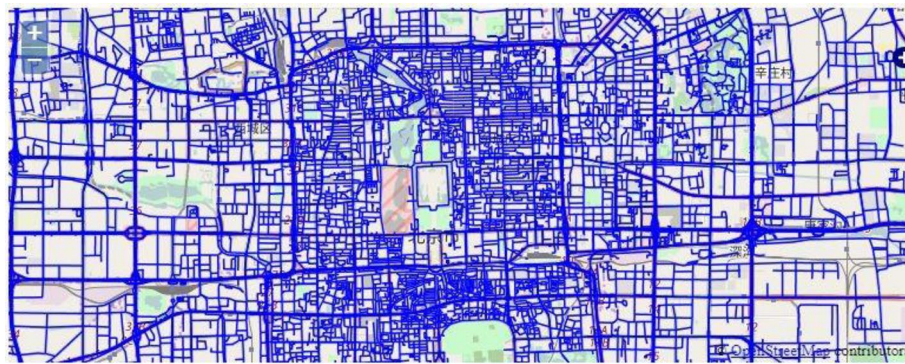


Fig. 6 OSM road network

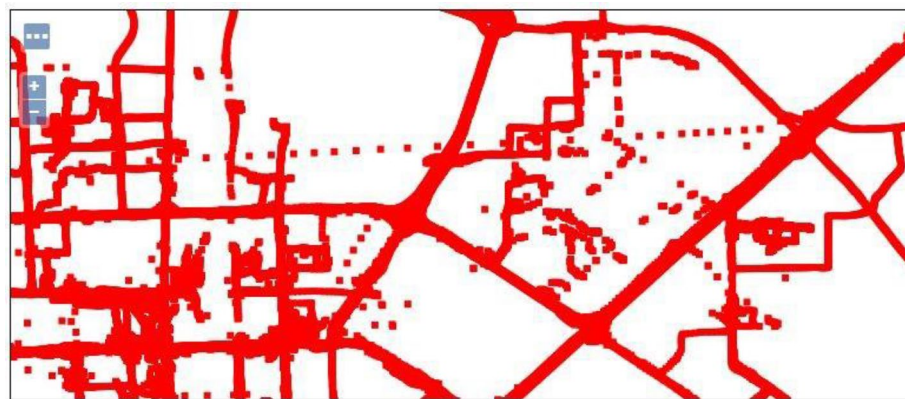
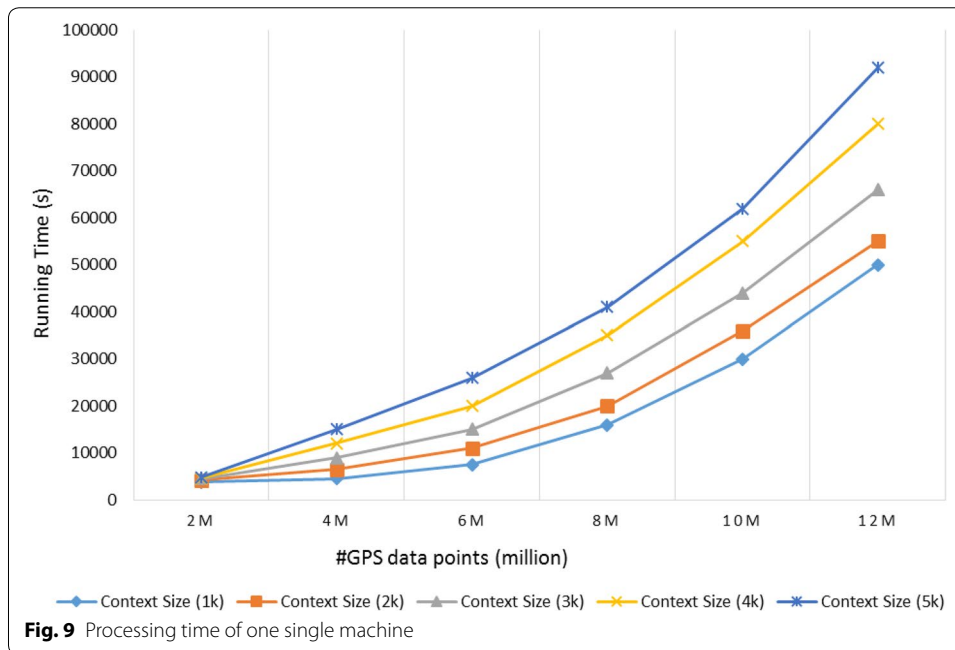
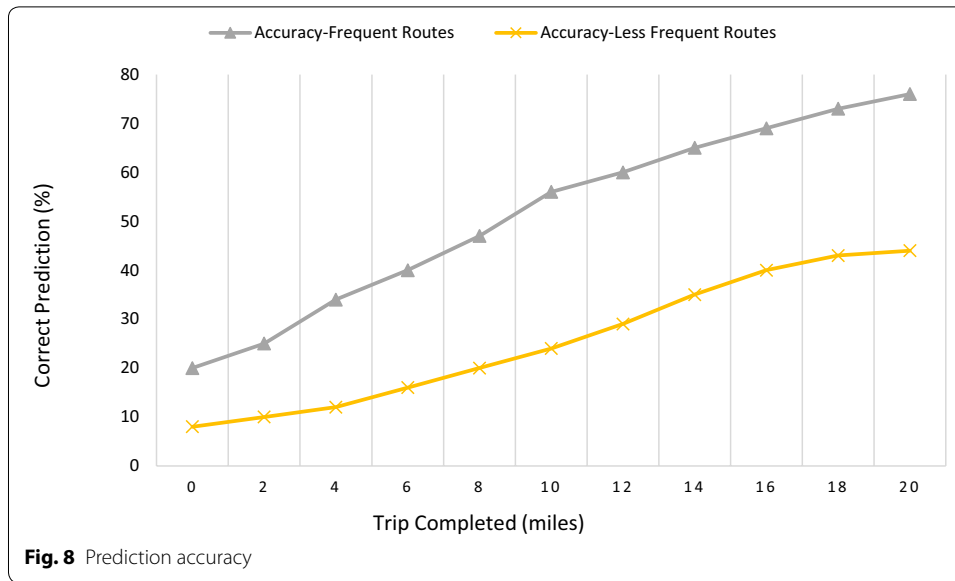


Fig. 7 Geolife trajectory sample data

convert OSM data into PostGIS compatible.sql files. The SQL data are then loaded to spatial database PostGIS. We used GeoServer tool for all data visualization. GeoServer supports easy connectivity to PostGIS database. A snapshot of the OSM Beijing road network is as shown in Fig. 6.

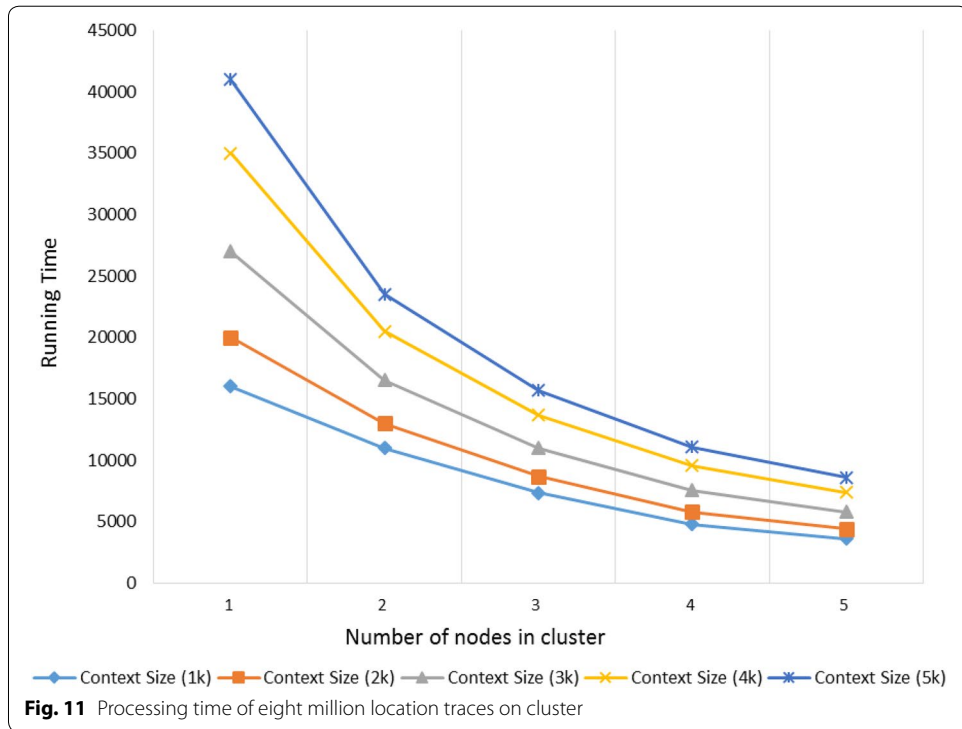
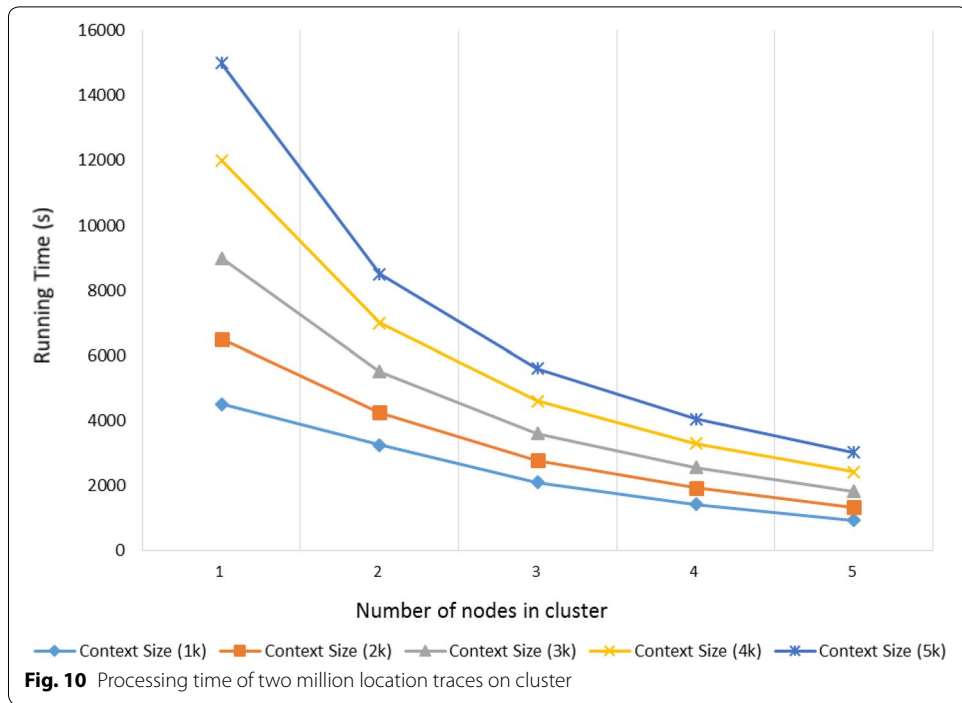
User location traces data

GPS data corpus used in this research work is from Geolife project. GPS data collection effort was made as Geolife project for the period 2007–2012 (Zheng et al. 2009; Zheng et al. 2008; Zheng et al. 2010). Geolife GPS data set contains time-stamped positional information of around 182 users. It contains around 17,621 trajectories which have 24,876,978 GPS data points. The length of all trajectories sums up to 1.2 million km and a total duration of around 48+ thousand hours. Devices used to capture data are GPS loggers as well GPS phones with different recording frequencies. Of all the trajectories, 91% trajectories have data collection frequency of every 1–5 s or 5–10 m per point and are dense data (Lammel 2008). Data collections were done from users while performing a variety of activities ranging from routine tasks like the movement from home to office and back to home as well other non-routine tasks

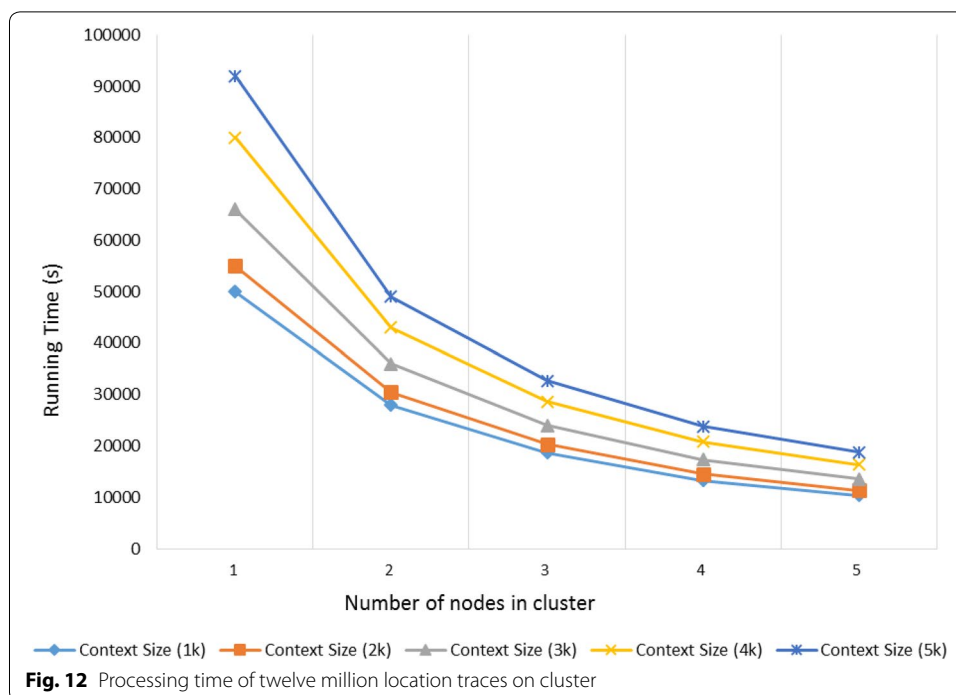


such as sightseeing, cycling, and shopping etc. (Zheng et al. 2009; Zheng et al. 2008; Zheng et al. 2010). Figure 7 shows GPS traces plotted from Geolife GPS data corpus.

Implementation and evaluation were performed in a cluster of distributed nodes which consisted of six compute nodes: one master and five worker nodes. Data were replicated with a factor of 5 to make sure that least time was spent on data transfer latency. Each independent node in the cluster had 8 GB internal memory and 64-bit processor with four cores. The prediction accuracy with the portion of trip



completed is shown in Fig. 8. Construction of the CTW tree on a single node is shown in Fig. 9. The CTW tree construction time on Hadoop cluster consisting of 2 million, 8 million and 12 million is shown in Figs. 10, 11 and 12, respectively.



Conclusion

In this work, the focus was on the construction of the PPM model in a distributed way from a huge corpus of GPS location traces. This model was then used for building a route prediction application. The application required road network data and GPS traces. Both data sets were sourced from openly available sources: road network data from OSM and GPS data from Geolife project. GPS location was decomposed into smaller units called user trips. User trips were map matched to road network to convert the data into a set of edges. This step is part of data preparation, which is a one-time activity. The map matching of GPS data to road network edges reduces the data size and makes the model construction faster than building a model from raw GPS data. For distributed construction, data were stored in HBase data store and MapReduce framework was used for computation. The design of processing was composed of two steps which are intuitive to implementation of MapReduce framework. The PPM model was constructed with the edges of the PPM tree annotated with the probability of their occurrence. The model was then used in the prediction of the route given a partial trajectory. We observed that the model construction phase is the most time consuming, but over distributed cluster processing the time decreases linearly with the addition of nodes in the cluster. Once the model is constructed, route prediction is not a time-consuming process, but is all about traversing a branch of a multiway rooted tree and is linear in search time. All tools and data sets used in this work are openly available in the public domain. All the snapshots presented in this work were taken during implementation from real data sets.

Authors' contributions

VST, SC and AA discussed the idea of PPM with respect to route prediction and its implementation aspects. VST and SC implemented the idea and contributed toward the first draft of the paper under the guidance of AA. AA and SC thoroughly proofread the manuscript and made all vital corrections. All authors read and approved the final manuscript.

Authors' information

Vishnu Shankar Tiwari is a postgraduate (master of technology—M Tech) in computer engineering from the Department of Computer Engineering, Indian Institute of Technology (IIT)-Bombay, Mumbai, India. He also holds an M Tech (computer applications) degree from the YMCA University of Science and Technology, India, and master of computer application (MCA) from the Maharshi Dayanand University, India. He currently works as Vice President in Technology at J.P. Morgan Chase & Co. with a total experience of more than 10 years in the teaching and software industry.

Arti Arya is Head of Department (HOD) and Professor at the Department of Computer Application, PES Institute of Technology, Bangalore South Campus. She holds Ph.D. in computer science from the Faculty of Technology and Engineering, Maharshi Dayanand University, India. She has an M Tech degree in computer science from the Allahabad Agricultural Institute, and master of science (mathematics) and bachelor of science (mathematics) from the Delhi University. Her areas of interests are spatial data mining, knowledge based systems, machine learning, artificial intelligence and data analysis. She has approximately 17 years of teaching experience (with 10 years of research) at the undergraduate and post graduate level. She is Senior Member IEEE, Life Member CSI and Life Member IAENG.

Sudha Chaturvedi is Lead Software Engineer in Aricent Technologies with more than 10 years of experience in the software industry and teaching/research. She holds master of technology—M Tech degree in computer engineering. Additionally, she holds master of computer applications (MCA) and master of business administration (MBA) degrees. She has expertise in implementation of machine learning models.

Acknowledgements

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Availability of data and materials

All data and material used is open source. Majorly, GPS data points are from GPS trajectory data set collected in (Microsoft Research Asia) Geolife project. Data set is made available for research from 2012 by Microsoft Research (<https://geotime.com/general/geolife-project/>). Map data used is from Open Street Map (OSM) which is an open project (<https://www.openstreetmap.org>).

Consent for publication

Authors consent the right to publish this article by Springer Open.

Ethics approval and consent to participate

This is author's own personal research work. Authors self-approves ethical approval and provide consent for participation.

Funding

This work is purely author's own work and authors own funding required for publishing of this research work.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Received: 3 April 2018 Accepted: 11 August 2018

Published online: 29 August 2018

References

- Begleiter R, El-Yaniv R, Yona G (2004) On prediction using variable order Markov models. *J Artif Intell Res* 22:385–421
- Bernstein D, Kornhauser A (1996) An introduction to Map Matching for personal navigation assistants, technical report, New Jersey TIDE Center Technical Report
- Celikeli E (2005) A cryptographic approach to language identification: PPM. In: Proceedings of the 7th Int'l conference on enterprise information systems (ICEIS)
- Chang F, Dean J, Ghemawat S, Hsieh WC, Wallach DA, Burrows M, Chandra T, Fikes A, Gruber RE (2008) Bigtable: a distributed storage system for structured data. *ACM Trans Comput Syst* 26(2):1–26. <https://doi.org/10.1145/1365815.1365816>
- Cleary J, Witten I (1984) Data compression using adaptive coding and partial string matching. *IEEE Trans Commun* 32(4):396–402. <https://doi.org/10.1109/tcom.1984.1096090>
- Cleary JG, Teahan WJ, Witten IH (1995) Unbounded length contexts for PPM. In: Storer JA, Cohn M (eds). Proceedings DCC '95, data compression conference: 28–0 Mar 1995. IEEE Computer Society Press, Snowbird, pp 52–61. <https://doi.org/10.1109/dcc.1995.515495>
- Dean J, Ghemawat S (2004) MapReduce: simplified data processing on large clusters. In: Proceedings of the 6th conference on symposium on operating systems design & implementation, December 06–08, 2004, San Francisco
- Effros M (2001) PPM performance with BWT complexity: a new method for lossless data compression. In: Proceedings of data compression conference, DCC 2000
- Froehlich J, Krumm J (2008) Route prediction from trip observations, society of automotive engineers (SAE) 2008 World Congress, April 2008, Paper 2008-01-0201
- Gilchrist J (2004) Parallel data compression with bzip2. In: Proc. of IASTED Intl. Conf. on Par. and Distrib. Computing and Sys. pp 559–564

- Greenfield JS (2002) Matching GPS observations to locations on a digital map. In: Proceedings of the 81st annual meeting of the transportation research board, Washington
- Hiroiyuki A, Kazuhiro K, Takashi I, Shigeichi H (2005) A PPM* algorithm using context mixture. In the Journal of IEIC, pp 35–40
- Joel R, Sirota V (2012) FPGA-based data compressor based on Prediction by Partial Matching. In: IEEE 27th convention of electrical and electronics engineers, Israel
- Lammel R (2008) Google's MapReduce Programming Model—revisited. *Sci Comput Program* 70:1–30
- Moffat A (1990) Implementing the PPM data compression scheme. *IEEE Trans Commun* 38(11):1917–1921. <https://doi.org/10.1109/26.61469>
- Quddus MA (2006) High integrity map-matching algorithms for advanced transport telematics applications, Ph.D. Thesis. Centre for Transport Studies, Imperial College London
- Quddus MA, Noland RB, Ochieng WY (2006) A high accuracy fuzzy logic based map matching algorithm for road transport. *J Intell Trans Syst* 10(3):103–115
- Schürmann T, Grassberger P (1996) Entropy estimation of symbol sequences. *Chaos* 6(3):414–427. <https://doi.org/10.1063/1.166191>
- Teahan WJ (1995) Probability estimation for PPM. In: Proceedings NZCSRSC'95
- Tiwari VS, Arya A, Chaturvedi SS (2013) Route Prediction using trip observations and map matching, advance computing conference (IACC). In: 2013 IEEE 3rd international, pp 583–587
- Tiwari VS, Arya A, Chaturvedi S (2014) Framework for horizontal scaling of map matching using MapReduce. In: IEEE, 13th international conference on information technology, ICIT 2014. <http://www.icit2014.in/>. Accessed 22–24 Dec 2014
- Zheng Y, Zhang L, Xie X, Ma W (2009) Mining interesting locations and travel sequences from GPS trajectories. In: Proceedings of international conference on World Wild Web (WWW 2009). ACM Press, Madrid Spain, pp 791–800.
- Zheng Y, Li Q, Chen Y, Xie X, Ma W (2008) Understanding mobility based on GPS data. In: Proceedings of ACM conference on Ubiquitous Computing (UbiComp 2008). ACM Press, Seoul, Korea, pp 312–321.
- Zheng Y, Xie X, Ma W (2010) GeoLife: a collaborative social networking service among user, location and trajectory. *IEEE Data Eng Bull* 33(2):32–40
- Zhou J, Golledge R (2006) A three-step map matching methods in GIS environment: travel/transport study perspective. *Int J Geogr Inf Syst*. X, X

Submit your manuscript to a SpringerOpen[®] journal and benefit from:

- Convenient online submission
- Rigorous peer review
- Open access: articles freely available online
- High visibility within the field
- Retaining the copyright to your article

Submit your next manuscript at ► springeropen.com
