Tiwari and Arya *Open Geospatial Data, Software and Standards* (2018) 3:10
https://doi.org/10.1186/s40965-018-0052-9

Open Geospatial Data,
Software and Standards

ORIGINAL ARTICLE Open Access

Distributed Context Tree Weighting (CTW) for route prediction

Vishnu Shankar Tiwari* and Arti Arya

Abstract

Route prediction play a vital role in many important location-based applications like resource prediction in grid computing, traffic congestion estimation, vehicular ad-hoc networks, travel recommendation etc. The goal of this work is to design scalable route prediction application based on Context Tree Weighting (CTW) modeling of user travel data. CTW is one of the widely used technique for text compression as well string sequence indexing and for prediction. CTW 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 horizontal scalability of CTW and addresses various challenges in distributed construction like reducing I/O, parallel computation of sequences and coming up with final CTW tree in a distributed environment efficiently. Map Reduce framework running over Hadoop file system is used for processing in distributed mode. Large GPS data set is map-matched to digitized road network obtained from Open Street Map and CTW model is built. A two-step construction of CTW tree is proposed which is implemented in the map-reduce framework. Horizontally scalable CTW model is built and evaluated for route prediction from a huge corpus of historical GPS traces.

Keywords: CTW, Route prediction, Scalable, Map reduce, Hadoop

Introduction

Route prediction is a key requirement in many location-based important applications like vehicular ad-hoc networks, traffic congestion estimation, resource prediction in grid computing, vehicular turn prediction, travel pattern similarity, pattern mining etc. Route prediction is a problem which deals with, given a sequence of road network graph edges already traveled by the user, predict the most probable edge of the network to be traveled. Our approach is to build a CTW model from a huge corpus of sequential trajectories traveled by the user in past. CTW model built is probabilistic in nature. Context tree weighting (CTW) is widely used in various applications in the area of data compression and machine learning [1]. Time-stamped GPS traces are collected over a long period of time. The chronological huge sequence of GPS traces is broken down into a smaller unit called trip [2, 3]. Trips are mapped to road network graph using map matching process which identifies the object's location on road network graph [4–6]. CTW 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 CTW tree based model and the most likely edge is found.

Willems et al. [7] presented CTW algorithm which is a strong lossless compression algorithm. Followed by this, many research work were carried out on CTW which were focused on achieving accuracy and reducing time complexity [8]. CTW models use a set of historical occurrences of sequences to predict the probability of which a specific symbol would appear at a given position in an input stream. CTW is a combination of many lower order Markov models. Real applications using CTW deals with processing of huge data sets in a sequential manner is time-consuming and is a hurdle in practical implementation. Existing techniques are designed for single machine and scalability is achieved by increasing system resources like processors, memory etc. on the single system [7, 9]. An alternative approach to achieve scalability is to make processes run

* Correspondence: vishnustiwari@gmail.com
Department of Computer Application, PES Institute of Technology, Bangalore South Campus, Bengaluru 560100, Karnataka, India

over a distributed cluster of independent systems. Construction of CTW in distributed mode still exists as a challenge. This work addressed this challenge by processing trips in parallel on distributed nodes and finally consolidating them to form CTW model. This is achieved in a two-step process. Set of user trips is decomposed into smaller sets and ported to compute 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 reducer node. Reducer consolidates the occurrences of various contexts and inserts in CTW tree. The final tree produced by reducer is CTW model and is used for route prediction. The major contribution of this work is a technique of distributed computation of CTW and its application in route prediction. All experiments and implementation are done on real datasets available openly in public domain.

Background

Context Tree Weighting (CTW) 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. CTW models use a set of historical occurrences of sequences to predict the probability of which a specific symbol would appear at a given position in an input stream. Arithmetic encoding was proposed in 1976 after which lots of statistical methods were proposed like PPM, PST etc. A strong technique known as Context Tree Weighting (CTW) was proposed after two decades by Willems et al. [7] which is a combination of many bounded length Markov models. Tjalkens et al. [10] proposed an encoding for CTW-method which was binary. It proposed to store the probabilities in the node of the CTW tree and lead to a reduction in storage space requirement. Sadakane et al. [11] presented a variant of CTW which established theoretical and experimental applications of CTW. Begleiter et al. [8] came up with a CTW implementation for supporting multi-alphabet scenario with a parameter to optimize CTW on binary alphabets. It was reported to achieve a compression rate of 2.27 bps on Calgary Corpus. Tjalkens et al. [12] extended CTW method for compressing ASCII and Byte files using binary decomposition and zero redundancy estimator. Volf [13] presented a variant of CTW which used a hierarchical tree based decomposition and applied for prediction over binary symbols. Each binary problem was solved by a slightly different variant of CTW. Begleiter et al. [1] further explored CTW and successfully applied in Artificial Intelligence (AI) applications including text prediction and music recognition and it worked well. Objectives of almost researchers on CTW were either improving its accuracy and execution on a single machine or its application in different fields of study. Comparison of major work in this area is summarized in Table 1. In spite of huge applicability, parallel execution of CTW model building is hardly explored. The idea of this work is to come up with a technique for distributed computation of CTW and its application in route prediction.

Methodology

CTW tree from user location traces

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), $(x_{t^1}, y_{t^1}, t^1)\ldots(x_{t^n}, y_{t^n}, t^n)$ which represents object's location (x_{t^k}, x_{t^k}) at time t^k. Chronological huge sequences of GPS traces are broken down in smaller units called trips [2, 3]. A user trip $T = (p_s, t_s, p_e, t_e)$ is an ordered sequence of GPS location data points $(p_i, t_i)\ \forall\ 1 \le i \le n$ where p_s, p_e are start and end positions and t_s, t_e are start and end time of trips respectively.

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

$$p_s = \left(x_{t^0}, y_{t^0}\right), t_s = t^0, p_e = \left(x_{t^m}, y_{t^m}\right), t_s = t^m$$

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

Trips are mapped to road network graph using map matching process which identifies the object's location on road network graph [14, 15]. Map matching is function f which for which input is GPS location and road

Table 1 Comparison of the most important algorithms for CTW construction

	Complexity	Parallel	Probabilistic
Begleiter et al. (2004) [1]	$O(n^2)$	No	yes
Willems et al. (1997) [7]	$O(n^2)$	No	yes
Tjalkens et al. (1997) [10]	$O(n^2)$	No	yes
Begleiter et al. (2006) [8]	$O(n^2)$	No	yes
Sadakane et al. (2000) [11]	$O(n^2)$	No	yes
Tjalkens et al. (1997) [12]	$O(n^2)$	No	yes
Volf, P. (2002) [13]	$O(n^2)$	No	yes
Aberg et al. (1997) [33]	$O(n^2)$	No	yes
Willems, F. (1998) [34]	$O(n^2)$	No	yes
Willems et al. (1996) [35]	$O(n^2)$	No	yes
Willems, F. (1996) [36]	$O(n^2)$	No	yes
Proposed CTW	$O(n^2)$	yes	yes

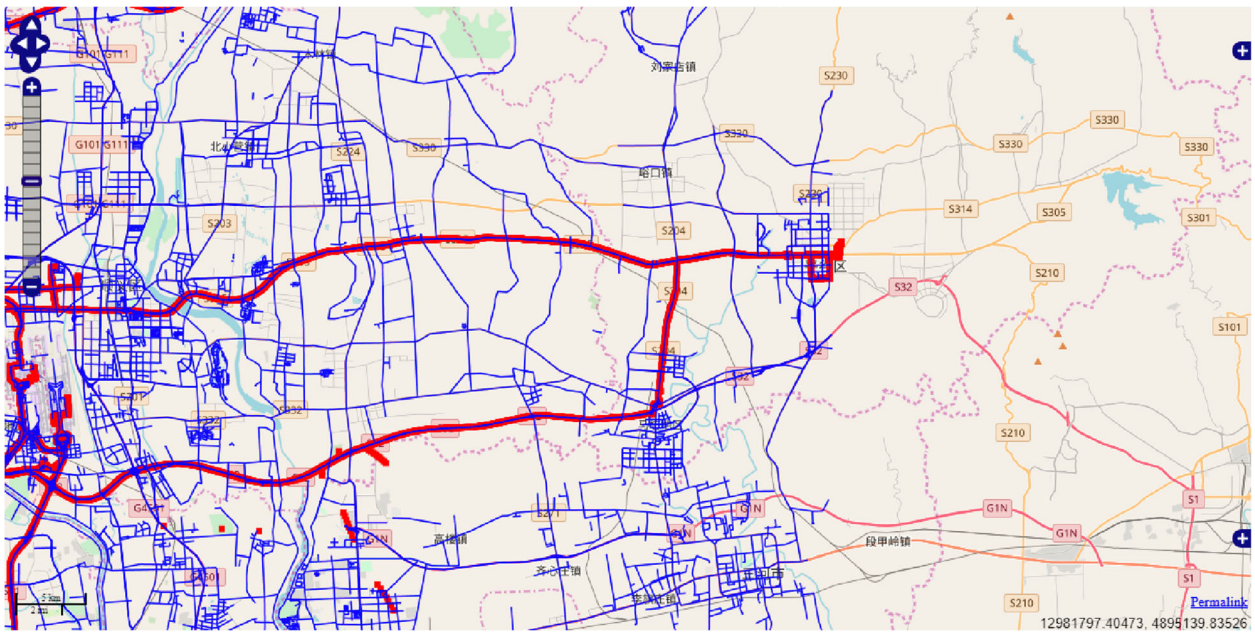

Fig. 1 User trip mapped to road network

network graph and 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 ordered sequence of road network edges. Let set $\Sigma = \{e_1, e_2, e_3, e_4, e_5\}$ be a finite set of all the edges of the digitized road network and Σ^* represents all finite length trips possible. Any trip user makes is essentially belongs to Σ^*. Let $X = e_0, e_1, \ldots, e_{n-1}$ with $x_i \in \Sigma$ & $X \in \Sigma^*$ be a trip then the length of the trip is given by $|X| = |e_0, e_1, \ldots, e_{n-1}|$.

For trip $X = e_1, e_2, e_3, e_4, e_5$ context for e_2 is e_1, for e_3 context is e_1, e_2 and context for e_4 is e_1, e_2, e_3, e_4 and so on. The ordered arrangement of all contexts $s\sigma$,where σ is symbol and s is context the of σ, is compact TRIE is known as the CTW tree [11]. For demonstration purpose let us assume road network edges set $\Sigma = \{e_1, e_2, e_3, e_4, e_5\}$ and a trip $X = e_1, e_2, e_5, e_1, e_3, e_1, e_4, e_1, e_2, e_5, e_1$. All the contexts with length $d = 2$ are as shown in Table 2 and resulting CTW tree is as shown in Fig. 2.

Two phase CTW tree construction

Proposed technique of CTW construction is a two-step process. The first phase is used to compute all contexts of length ≤ d where d denotes the length of trip (number of edges in the trip). All the contexts $s\sigma$ is generated for each symbol σ and are put into a map which stores sequence as key and frequency as value. Second phase, consolidates the occurrences of various contexts and inserts in a context tree. Final tree produced by reducer is CTW model. Both steps are discussed below. In next section, two phase implementation is extended to execute over map reduce framework.

Processing is as summarized in Algorithm I below. 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 $d \leq 2$

Table 2 All contexts of length (D) ≤ 2 for $e_1, e_2, e_5, e_1, e_3, e_1, e_4, e_1, e_2, e_5, e_1$

S.No.	D	Context(s)	Symbol(σ)	sσ
1	1	e_1	e_2	e_1, e_2
2	1	e_2	e_5	e_2, e_5
3	1	e_5	e_1	e_5, e_1
4	1	e_1	e_3	e_1, e_3
5	1	e_3	e_1	e_3, e_1
6	1	e_1	e_4	e_1, e_4
7	1	e_4	e_1	e_4, e_1
8	2	e_1, e_2	e_5	e_1, e_2, e_5
9	2	e_2, e_5	e_1	e_2, e_5, e_1
10	2	e_5, e_1	e_3	e_5, e_1, e_3
11	2	e_1, e_3	e_1	e_1, e_3, e_1
12	2	e_3, e_1	e_4	e_3, e_1, e_4
13	2	e_1, e_4	e_1	e_1, e_4, e_1
14	2	e_4, e_1	e_2	e_4, e_1, e_2

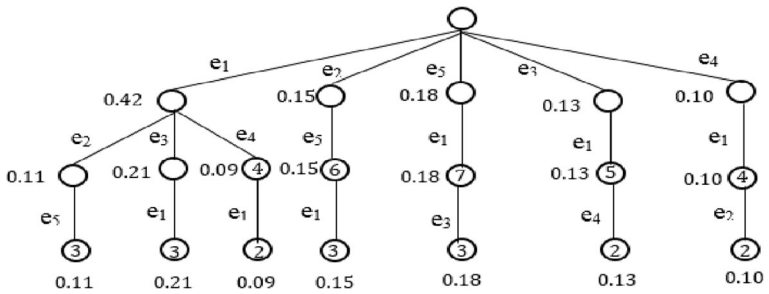

Fig. 2 CTW tree construction

alongwith target symbol σ denoted by $s\sigma$ computed by Algorithm I is as shown in Table 3.

Algorithm I: CTW Context and frequency calculation
Input: String X and context length d = 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 $|S'| \leq d+1$:
   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 $M'(S')$

Length of string X is denoted by n. All contexts of length d in X can be calculated in linear time $\Theta(n^2)$ by scanning X from left to right by maintaing a window of size d. Window is advaced by one unit on scanning one symbol. Maximum number of context strings each of length $\leq d$ that can appear in map is $\Theta(n^2)$ where $d \ll n$. This can happen only if contexts does not overlap otherwise in practice number of contexts$\leq\Theta(n)$.

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 the tree is found which is matching/overlapping with current context then for all the nodes in an overlapping path is increased by the frequency of occurring and for remaining nodes are inserted starting the end of the overlapping path. All contexts computed by Algorithm II is as shown in Table 3.

Table 3 All contexts of length (d) ≤ 2 for e_1, e_2, e_5, e_1, e_3, e_1, e_4, e_1, e_2, e_5, e_1 with frequency

S.No.	Sσ	Frequency(f)
1	e_1, e_2	2
2	e_2, e_5	2
3	e_5, e_1	2
4	e_1, e_3	1
5	e_3, e_1	1
6	e_1, e_4	1
7	e_4, e_1	1
8	e_1, e_2, e_5	2
9	e_2, e_5, e_1	2
10	e_5, e_1, e_3	1
11	e_1, e_3, e_1	1
12	e_3, e_1, e_4	1
13	e_1, e_4, e_1	1
14	e_4, e_1, e_2	1

Algorithm II: CTW Tree Construction
Input: Map M' of all contexts including target symbol σ with frequency $f(context, count)$
Output: CTW Tree with frequency
Algorithm:
1. Instantiate an empty tree T
2. for each $< key, value >$ pair $< context_i, count_i > \in M'(context, count)$
   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

The height of the tree is $h = \Theta(d+1) \cong \Theta(d)$ is linear of the length of context. All branches are of equal length and length of each branch is necessarily $\Theta(d)$. As established earlier, Maximum number of context strings each of length d that can appear in the map is $\Theta(n-d) \cong \Theta(n)$ when $d \ll n$. As soon as d approaches n then the total number of context string aproaccches $O(1)$. In practice, $d \ll n$ means d is way less than n and nearly constant $O(1)$ so we assume a maximum number of context string $\Theta(n)$ without loss of generality. Iteratively each string $s\sigma$ of length $|s\sigma| = d$ which is formed by string concatenation of context s and target symbol σ, is inserted into CTW tree (starting with the empty tree). Thoretically, cost of each insertion is $O(d)$. Number of such insertions is $\Theta(n)$ leads to total cost of $O(nd) \cong O(n^2)$. In actual practice, it is very likely that pattern repeats and contexts are same. In such a scenario, a number of total context strings $n(s\sigma) \ll \Theta(n)$ and cost of construction of CTW tree on a single machine from the output of Algorithm I is $\ll\Theta(nd)$. But as stated, d is way less than n and is approximately a constant and hence complexity of algorithm approaches $\Theta(n)$. Combining running time of both phases is $\Theta(2n) = \Theta(n)$.

Distributed construction of CTW tree

In order to achieve distributed construction of CTW tree-based model, two-step process described in the earlier section is extended to execute over Hadoop cluster

leveraging the map-reduce computation framework. The first phase is executed by mapper module. Map matches GPS traces and decomposed in smaller units called trips are portioned into chunks of a set of trips and to mapper module. All the contexts $s\sigma$ is generated by 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 mapper for computation of contexts under map reduce model is described in Algorithm III.

To demonstrate the distributed construction of CTW tree we take a string below. This will be used as running example throughout for further discussion.

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

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

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

$$split\ S_2 = e_5, e_1, e_3, e_1, e_4, e_1, e_2, e_5, e_1, e_3, e_1$$
$$processed\ by\ mapper\ m_2$$

Split S_1 has span (1 to 11) and split S_2 has span (10 to 20). Output of both mappers $m_1$and m_2 are summarized in Tables 4 and 5 respectively. In this example, context $s\sigma$ serves as key and frequency (f) as value.

Table 4 All contexts of length (d) ≤ 2 for e_1, e_2, e_5, e_1, e_3, e_1, e_4, e_1, e_2, e_5, e_1 with frequency computed by m_1

S.No.	sσ	Frequency(f)
1	e_1, e_2	2
2	e_2, e_5	2
3	e_5, e_1	2
4	e_1, e_3	1
5	e_3, e_1	1
6	e_1, e_4	1
7	e_4, e_1	1
8	e_1, e_2, e_5	2
9	e_2, e_5, e_1	2
10	e_5, e_1, e_3	1
11	e_1, e_3, e_1	1
12	e_3, e_1, e_4	1
13	e_1, e_4, e_1	1
14	e_4, e_1, e_2	1

Table 5 All contexts of length (d) ≤ 2 for e_5, e_1, e_3, e_1, e_4, e_1, e_2, e_5, e_1, e_3, e_1 with frequency computed by m_2

S.No.	sσ	Frequency(f)
1	e_5, e_1	2
2	e_1, e_3	2
3	e_3, e_1	2
4	e_1, e_4	1
5	e_4, e_1	1
6	e_1, e_2	1
7	e_2, e_5	1
8	e_5, e_1, e_3	2
9	e_1, e_3, e_1	2
10	e_3, e_1, e_4	1
11	e_1, e_4, e_1	1
12	e_4, e_1, e_2	1
13	e_1, e_2, e_5	1
14	e_2, e_5, e_1	1

Algorithm III - Mapper function: Context and frequency calculation of local partition
Input: Split chunk S' of original data string S and context length d = 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 $M'(S')$
3. Send data map $M'$to reducer function

The output of mapper modules as a set of key-value pairs where the key is context and value as the frequency is emitted as input to the reducer. Reducer is the common point where intermediate key-value pairs computed by each mapper is emitted. Even before sending the output to the reducer, the framework does a consolidation by adding the frequencies for each context as key. For example if from one mapper value received is <e_1, e_2 | 4 > and < e_1, e_2 | 10 > then after merge the final entry becomes <e_1, e_2 | 14>. Each key-value pair is unique is ensured during this step. If multiple entries exist for the same key then consolidation is done before sending it to reducer [16, 17]. If data does not fit into memory then it is periodically written to disk [18]. The result of consolidation is shown in Table 6. Final CTW tree construction is taken care by reducer function. 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 the tree is found which is matching/overlapping with current context then for all the nodes in an overlapping path is increased by the frequency of occurring and for remaining nodes are inserted starting the end of the overlapping path. Implementation of the reducer is as described in Algorithm IV and output of reducer is in Table 7. CTW tree thus constructed is shown in Fig. 2.

Table 6 Result of merging of intermediate key/value pairs by Map-Reduce framework

S.No.	Key(k)	Frequencies	<K,<value_list>>
1	e_1, e_2	2, 1	< e_1, e_2, < 2, 1 > >
2	e_2, e_5	2, 1	< e_2, e_5, < 2,1 > >
3	e_5, e_1	2, 2	< e_5, e_1, < 2, 2 > >
4	e_1, e_3	1, 2	< e_1, e_3, < 1, 2 > >
5	e_3, e_1	1, 2	< e_3, e_1, < 1, 2 > >
6	e_1, e_4	1, 1	< e_1, e_4, < 1, 1 > >
7	e_4, e_1	1, 1	< e_4, e_1, < 1, 1 > >
8	e_1, e_2, e_5	2, 1	< e_1, e_2, e_5, < 2, 1 > >
9	e_2, e_5, e_1	2, 1	< e_2, e_5, e_1, < 2,1 > >
10	e_5, e_1, e_3	1, 2	<e_5, e_1, e_3, < 1, 2 > >
11	e_1, e_3, e_1	1, 2	< e_1, e_3, e_1, < 1, 2 > >
12	e_3, e_1, e_4	1, 1	< e_3, e_1, e_4, < 1, 1 > >
13	e_1, e_4, e_1	1, 1	< e_1, e_4, e_1, < 1, 1 > >
14	e_4, e_1, e_2	1, 1	< e_4, e_1, e_2, < 1, 1 > >

Algorithm IV - Reducer function: CTW Tree Construction
Input: Maps of suffixes with frequency from mapper modules $M'_z(suffix, count)$ ..... $M'_{z+k}(suffix, count)$
Output: CTW Tree with frequency
Algorithm:
1. Instantiate an empty tree T
2. For each map $M'_i \in M'_{z+k}(suffix, count) \forall 0 \le j \le k$
   For each $suffix_i, count_i \in M'_i$
   I. Traverse tree till any branch has overlap with $suffix_i$ and increment each node by $count_i$
   II. For remaining alphabets of $suffix_i$ 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 CTW

The objective is to predict the next edge σ E on the road network given the user traveled trajectory $S = (x_{t^0}, y_{t^0}, t^0), (x_{t^1}, y_{t^1}, t^1) \ldots (x_{t^n}, y_{t^n}, t^n)$ based on information learnt from historical user travel data. In order to predict next edge σ, S is map matched to digitized road network using map matching process f as described in earlier sections.

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

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

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

p is the conditional probability of occurrence of σ given the event $e_i e_{i+1} \ldots e_{i+n}$ has already occurred. CTW tree constructed has information learnt from historical travel data of the user. Since CTW is an unbounded Markov model, the corresponding tree may be balanced and each path from root may be of different length. This makes CTW a variable order Markov model. In the worst case one have to traverse the longest branch of the CTW tree. If the length of the longest branch of tree is k then the complexity of prediction using CTW tree is $O(k)$. Probabilities of occurrence of each node starting with root node is as shown in Fig. 2. Route prediction function denoted by a function *Route_Predict* can be represented as:

$$Route\_Predict(e_i e_{i+1} \ldots e_{i+n}) \rightarrow \sigma$$

Below cases demonstrates prediction *Route_Predict* function over CTW model constructed by Algorithm IV.

Case I

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

$$\begin{aligned} \mathrm{p}(e_1 | \varepsilon) = \frac{16}{38} = 0.42, \quad \mathrm{p}(e_2 | \varepsilon) = \frac{6}{38} = 0.15, \quad \mathrm{p}(e_3 | \varepsilon) \\ = \frac{5}{38} = 0.13, \quad \mathrm{p}(e_4 | \varepsilon) = \frac{4}{38} = 0.10, \quad \mathrm{p}(e_5 | \varepsilon) \\ = \frac{7}{38} = 0.18 \end{aligned}$$

Hence $Route\_Predict(\varepsilon) \rightarrow e_1$.

Case II

Another case we explore is when edge e_2 is traversed so far $S = . e_2$. Length of input trajectory is 1 unit only and consists of single edge. Candidate edge after e_2 is already traversed is only one and is e_5. In this case probability of occurrence of e_5 after e_2 as context is $\mathrm{p}(e_5 | e_2) = 1$. Hence $Route\_Predict(e_2) \rightarrow e_5$.

Table 7 Result of calculation of the sum of value list associated with keys

S.No.	Key(k)	<K, Sum(Values)>
1	e_1, e_2	< e_1, e_2, < 3 > >
2	e_2, e_5	< e_2, e_5, < 3 > >
3	e_5, e_1	< e_5, e_1, < 4 > >
4	e_1, e_3	< e_1, e_3, < 3 > >
5	e_3, e_1	< e_3, e_1, < 3 > >
6	e_1, e_4	< e_1, e_4, < 2 > >
7	e_4, e_1	< e_4, e_1, < 2 > >
8	e_1, e_2, e_5	< e_1, e_2, e_5, < 3 > >
9	e_2, e_5, e_1	< e_2, e_3, e_1, < 3 > >
10	e_5, e_1, e_3	<e_5, e_1, e_3, < 3 > >
11	e_1, e_3, e_1	< e_1, e_3, e_1, < 3 > >
12	e_3, e_1, e_4	< e_3, e_1, e_4, < 2 > >
13	e_1, e_4, e_1	< e_1, e_4, e_1, < 2 > >
14	e_4, e_1, e_2	< e_4, e_1, e_2, < 2 > >

Case III

Next case is when input trajectory is $S = \{e_1\}$ and only one edge e_1 is traversed so far. But multiple candidates ($\{e_2, e_3, e_4\}$) are there having event of traveling over e_1 has occurred. Probabilities of each candidate is as follows:

$$p(e_2|\, e_1) = \frac{5}{18}, p(e_3|\, e_1) = \frac{9}{18}, p(e_4|\, e_1) = \frac{4}{18}$$

Hence $Route\_Predict(e_1) \rightarrow e_3$.

Case IV

Next we consider a case when multiple edges are traveled and input to *Route_Predict* function is $\{e_1, e_2\}$. Possible candidate for travel next is edge e_5 having said event of traveling over $\{e_1, e_2\}$ has 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 is not yet seen by CTW model. For example, if the user has traveled path $\{e_3, e_4\}$ but in CTW tree no such path exists means this something which has not occurred in past. Hence prediction function result is $Route\_Predict(e_3, e_4) \rightarrow \varepsilon$. This can happen when user has reached the destination and there is nothing to predict and in another case, it's a new route. In later case, new routes when, found should be sent to model for learning.

Case VI

All scenarios described above predicts one next hop edge. It is possible to predict multiple edges too. For example from root node $p(e_1|\, \varepsilon) = \frac{16}{38} = 0.42$ is the highest among all available candidates. For e_1 next edge with the highest probability is e_3 with a probability of $p(e_3|\, e_1) = \frac{9}{18} = 0.50$.

Results and discussion

Map data and GPS location traces are two data sets required for implementation of proposed CTW based route prediction. Data sets used are described below.

Digitized road network data is used for converting user location GPS traces into an ordered set of edges. Road network graph consists of two kinds of attributes:

1. Non-spatial features like width, length, speed, name and turn restriction of the road represented by an edge in the graph.
2. Spatial data that represents the geometry of road network.

Both of these are sourced from Open Street Map (OSM). OSM is user-generated maps platform where crowdsourced model of data building is used. Open Street maps creates and provides free geographic data such as street maps and is published under an open content license, with the intention of promoting the free use and re-distribution of the data (both commercial and non-commercial) (https://www.openstreetmap.org), [19]. OSM has evolved as the biggest open map data platform which is being used in research as well practical applications. We mention few of cases here but is not limited to and application area is continuously growing. Mumbai Navigator is similar experimental travel planning program developed at IIT-Bombay for the city of Mumbai [20]. On selecting the origin and destination of travel from the list, produces a travel plan, with an estimate of the total journey time including the waiting time. The plan may be a multi-modal trip which may require one to change buses or trains or travel by walk. It uses the graphic map as the base and the line features representing the roads, stored in the Spatial Database. This project is a proof of concept for using the spatial data stored in spatial databases for Route planning. Rousell et al. [21] used OSM data to extract landmarks. Tags in OSM data like shops, station etc. are used upfront to determine the landmarks and also other potential features like way turn point which can identify landmark is also explored in detail. Experimentation used real data sets and established successful extraction of landmarks. Navigation and mobility is the most effective use of OSM. Zipf et al. [22] and Mobasheri et al. [23] explored the suitability of OSM data for use by people with limited mobility. Two aspects, graph network and routing engine are two major components for such a system. As we stated, OSM provides graph network for visualization as well road network data which can be stored in spatial databases like PostGIS for development of routing engines. For applications like wheelchair routing, much granular level data in a limited area is required like sidewalks etc. Data quality is a major concern in such an application. As we observed in this work that geographical data set is huge and for practical realization, a parallel processing platform is required. Travel recommendation is one of the top explored area in the research community. Most of the research uses GPS traces to achieve this. Sun et al. [24] used geotagged photos to identify the location and routing. They used OSM data for routing and travel recommendation. Bakillah et al. [25] realized this and explored the existing techniques in area Big Data for their applicability in volunteered geographic information (VGI). Based on Big VGI data routing issues are explored and various challenges are addressed. The best information of any area can be gathered from local residents. Such a specialized information can be very helpful in disaster management applications. Haworth et al. [26] explored the application of OSM data in disaster

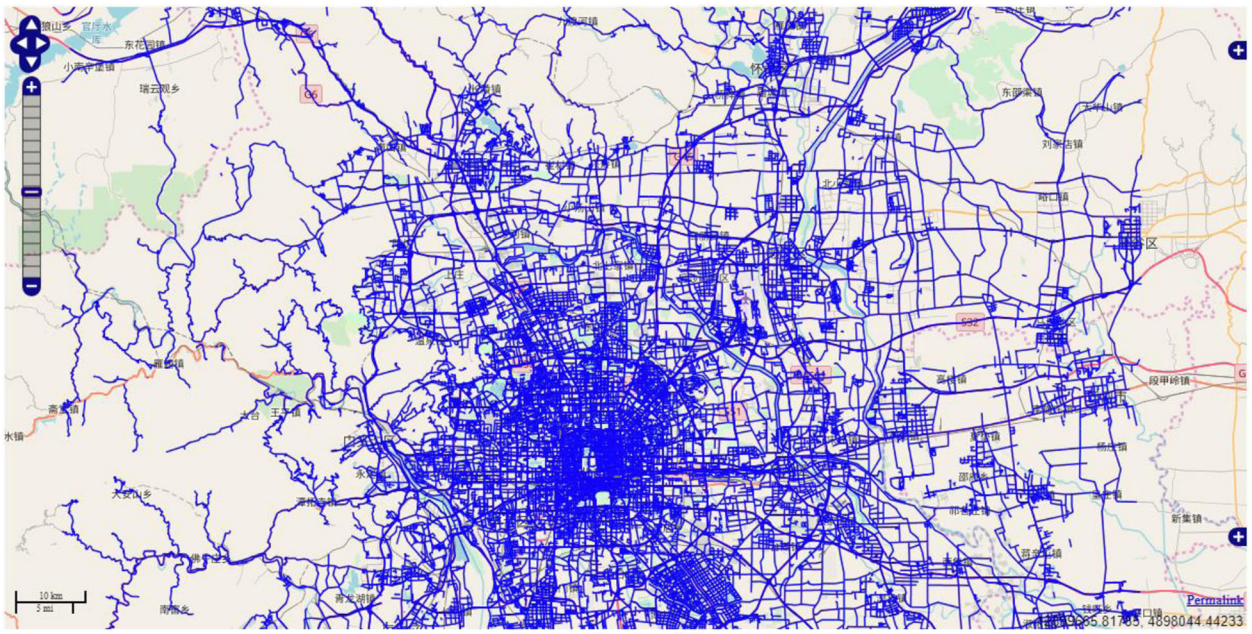

Fig. 3 Snapshot of road network data extracted from OSM

management scenarios. Zook et al. [27] established usefulness of OSM data in relief during earthquake situation. OSM is makes map data in a digitized format which makes web-based mapping services feasible. Using this facility even without physically present, individuals can make a considerable difference in relief and aid agencies work. It is interesting to observe how OSM can help in achieving a better environment. OSM not only helps vehicle routing but with the granular data available even routing for bicycles too can be done. Sun et al. [28] explored various factors which influences usage of bicycle sharing system. All routing data visualization and analysis was done using OSM data. OSM provides various kinds of GIS data like political boundaries, land use data, water bodies and road network data etc. (https://www.openstreetmap.org). Campus GIS is a project

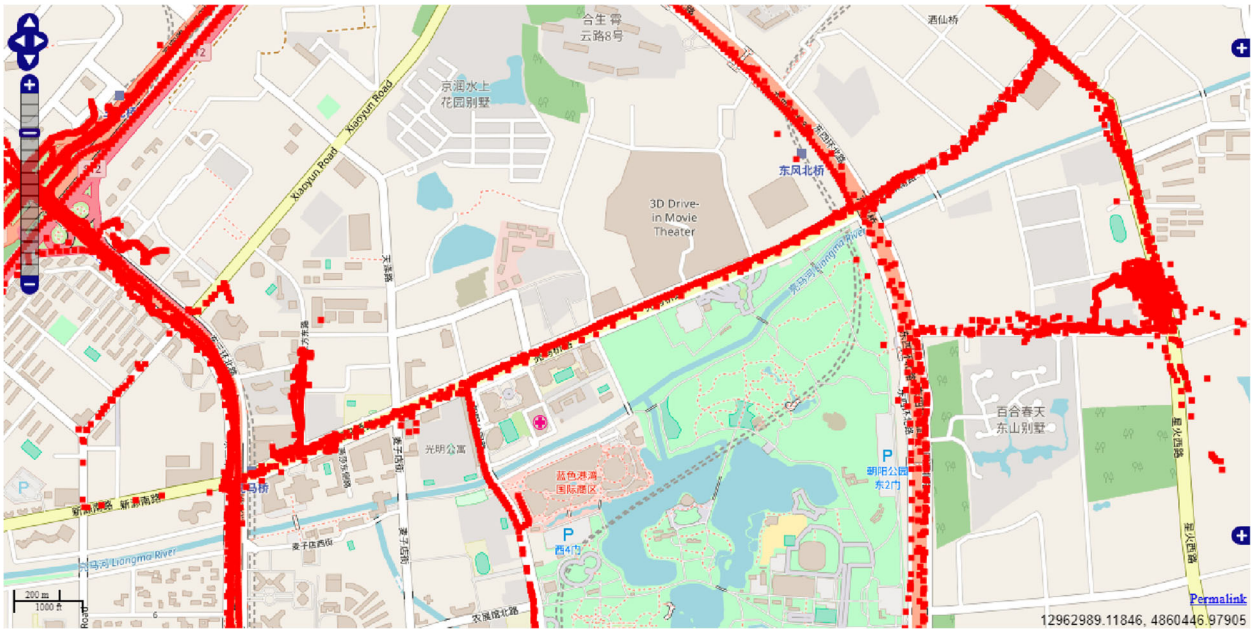

Fig. 4 Snapshot of GPS data from Geolife

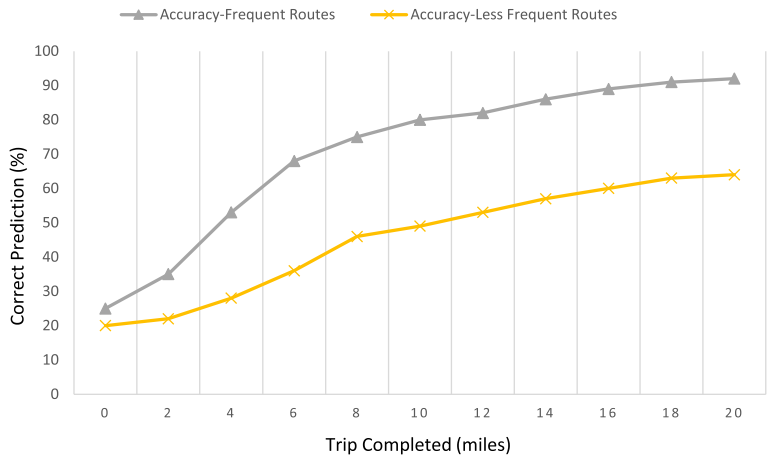

Fig. 5 Prediction accuracy in terms of miles of trip completed

developed at GISE Lab, Department of Computer Science, IIT Bombay. This project Proposed to store the spatial data in Spatial Enabled Databases. These databases can be queried to view the resources like Buildings, electrical resources, free land and used land etc. present in the campus. This Project is a demonstration of showing multiple data layers of IIT Bombay over Web using OSM [29]. Data is made available under open content license with the intention to promote free use. We used only road network data from OSM. Data can be downloaded using OSM interface (https://www.openstreetmap.org) if the area is smaller. It depends upon how much data is contained in a selected rectangle- in urban areas it may be up to just a kilometer whereas in rural areas, it can be several kilometers. If the area is larger, data can be downloaded from the official download page of OSM (https://planet.openstreetmap.org/). Data is available in various standard formats like image (.jpg, .png etc.) or XML format which comes with extension .osm. We used an .osm format which we parsed using open source tool called Osm2pgsql (wiki.openstreetmap.org/wiki/Osm2pgsql). It is used to convert OSM data into PostGIS compatible .sql files. SQL data is them loaded to spatial database PostGIS. We used GeoServer tool for all data visualization. GeoServer supports easy connectivity to PostGIS database. Snapshot of OSM Beijing road network is as shown in Fig. 3.

GPS data corpus used in this research work is from Geolife project. GPS data collection effort was made as Geolife project for the period from the year 2007 to 2012. Geolife GPS dataset contains time-stamped positional information of around 182 users. It contains around 17,621 trajectories which have 24,876,978 GPS data points. Length of all trajectories sums up to 1.2 million kilometers and total duration of around 48+ thousand hours. A device used to capture data were 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 (https://www.microsoft.com/en-us/research/publication/

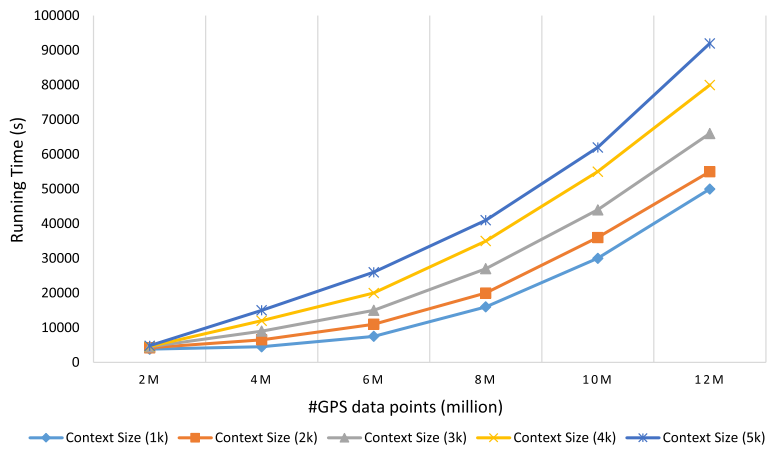

Fig. 6 Processing time of on single machine

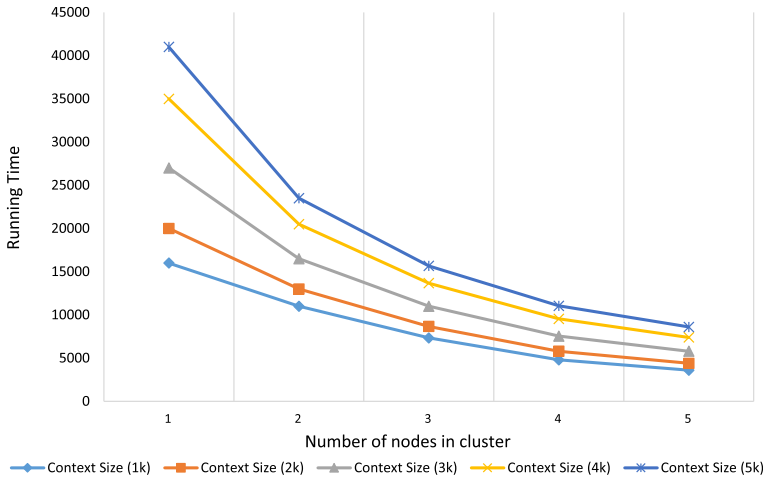

Fig. 7 Processing time of on cluster

geolife-gps-trajectory-dataset-user-guide). 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 site seeing, cycling, and shopping etc. Geolife GPS data trajectory can be download from below link (https://www.microsoft.com/en-us/research/publication/geolife-gps-trajectory-dataset-user-guide/). Figure 4 shows GPS traces plotted from Geolife GPS data corpus.

OSM road network data was loaded into PostGIS database. GPS location traces downloaded from Geolife project is flat file format was processed and loaded into PostGIS database. PostGIS is used as staging area for all data storage needs. Visualization of both GPS traces as well road network data, open source tool is known as Geoserver was used. GeoServer has the capability of sourcing data from PostGIS data and can render data over an HTTP connection. We moved data to and fro between PostGIS and HDFS for distributed processing of data. On Hadoop file system data was stored in columnar data store known as HBase. During CTW model training phase data was sourced from HBase for distributed processing as that was the most time-consuming process and is a bottleneck in practical implementation. After CTW tree constructed, the processed model was brought to PostGIS. In route prediction phase and data visualization, PostGIS database was used as data source. Implementation and evaluation were performed in a cluster of distributed nodes which consisted of 6 compute nodes: one master and 5 worker nodes. Data was replicated with a factor or 5 to make sure least time is spent in data transfer latency. Each independent node in the cluster had 8 GB internal memory and 64-bit processor with 4 cores. Prediction accuracy with a portion of trip completed is shown in Fig. 5. Construction of CTW tree on a single node is shown in Fig. 6. CTW tree construction time on Hadoop cluster consisting of 5 nodes computing 8 million GPS traces is shown in Fig. 7. Table 8 shows the most important milestones in the area of route prediction. Summarization is based on the methodology used, horizontal scalability and accuracy. The achievement of current work is- it is horizontally scalable yet competes with state of art methodologies for route prediction.

Conclusions and future work

In this work, the focus was on the construction of CTW model in the distributed way from a huge corpus of GPS location traces. GPS location was decomposed into smaller units called user trips. User trips were map-matched

Table 8 Comparision with most important researches in route prediction

Authors	Method for prediction	Horizontal scalability	Rate
Proposed CTW Route Prediction	Context Tree Weighting (CTW)	Yes	40–90%
Simmons et al. (2006) [37]	Hidden Markov Model	No	70–80%
Burbey et al. (2008) [38]	Prediction by Partial Match (PPM)	No	92%
Tiwari et al. (2012) [39]	Closest Match Algorithm	No	40%
Lung et al. (2014) [40]	Hidden Markov Model	No	68.3%
Neto et al. (2015) [41]	Prediction by Partial Match (PPM)	No	46%
Amirat et al. (2017) [42]	Graph Mining	No	76%
Froehlich et al. (2008) [3]	Closest match	No	40–90%
Tiwari et al. (2017) [43]	Probabilistic Suffix Tree	Yes	85

to road network to convert the data into a set of edges. The map matching of GPS data to road network edges reduces the data size and make model construction faster than building model from raw GPS data. CTW model was constructed with edges of CTW tree annotated with probability of their occurrence. The model was then used in prediction of the route given a partial trajectory. We observed that model construction phase is the most time consuming but over distributed cluster processing time decreases linearly with the addition of nodes in the cluster. Once the model is constructed, route prediction is not a time-consuming process. It is important to note that quality of data used in such a system really matters. OSM is a crowdsourced data and data quality is a major concern [30]. However, lots of research is in progress in this area and should be considered for future work [31, 32].

Acknowledgements
We thank Geospatial Information Science and Engineering (GISE) Lab, Indian Institute of Technology, IIT-Bombay, India for carrying out some initial part of the work at their lab. We would also like to thank anonymous reviewers, whom reviews helped us to bring this manuscript to the current form.

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

Availability of data and materials
All data and material used is open source. Majorly, GPS data points are from GPS trajectory dataset collected in (Microsoft Research Asia) Geolife project. Dataset 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. (www.openstreetmap.org).

Authors' contributions
VST and AA discussed the idea of CTW with respect to Route Prediction and its implementation aspects. VST has implemented the idea and contributed towards the first draft of the paper under the guidance of AA. AA thoroughly proofread the manuscript and made all vital corrections. Both the authors have read and finally approved the manuscript.

Authors' information
Vishnu Shankar Tiwari is a post graduate (Master of Technology- M.Tech.) in Computer Engineering from Department of Computer Engineering, Indian Institute of Technology (IIT)-Bombay, Mumbai, India. Also, holds M.Tech. (Computer Applications) from YMCA University of Science and Technology, India and Master of Computer Application (MCA) from Maharshi Dayanand University, India. Working in software industry for more than 8 years.
Arti Arya is Head of Department (HOD) and Professor at Department of Computer Application, PES Institute of Technology, Bangalore South Campus. She holds Ph.D in Computer Science from Faculty of Technology and Engineering, Maharshi Dayanand University, India. She has M.Tech in Computer Science from Allahabad Agricultural Institute, Master of Science (Mathematics) and Bachelor of Science (Mathematics) from Delhi University. Her areas of interests are Spatial Data Mining, Knowledge based systems, Machine Learning, Artificial Intelligence, Data Analysis. She has approx. 17 years of teaching experience (of which 10 years of research) at Undergraduate and Post Graduate level. She is Senior Member IEEE, Life Member CSI and Life Member IAENG.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note

Received: 3 March 2018 Accepted: 11 May 2018
Published online: 02 July 2018

References

1. Begleiter R, El-Yaniv R, Yona G. On prediction using variable order Markov models. J Artif Intell Res. 2004;22:385–21.
2. Tiwari VS, Arya A, Chaturvedi S. Framework for horizontal scaling of map matching using map-reduce. In: IEEE, 13th International Conference on Information Technology, ICIT 2014; 2014.
3. Froehlich J, Krumm J. Route prediction from trip observations, Society of Automotive Engineers (SAE) 2008 world congress, paper 2008–01-0201. 2008.
4. Liu Y, Li Z. A novel algorithm of low sampling rate GPS trajectories on map-matching. EURASIP J Wirel Commun Netw. 2017; 2017:30. https://link.springer.com/article/10.1186/s13638-017-0814-6.
5. Zhou J, Golledge R. A three-step general map matching method in the GIS environment: travel/transportation study perspective. Int J Geogr Inf Syst. 2006;8(3)243–60. https://scholar.google.com/scholar_lookup?title=A%20three-step%20general%20map%20matching%20method%20in%20the%20GIS%20environment%3A%20travel%2Ftransportation%20study%20perspective&author=J.%20Zhou&author=R.%20Golledge&journal=260&publication_year=2006.
6. Manikandan R, Latha R, Ambethraj C. An analysis of map matching algorithm for recent intelligent transport system. Asian J Appl Sci. 2017; 05(01) (ISSN: 2321 – 0893). https://www.ajouronline.com/index.php/AJAS/article/view/4642.
7. Willems F, Shtarkov Y, Tjalkens T. Reflections on The context-tree weighting method: basic properties. Newsl IEEE Inf Theory Soc. 1997; http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.109.1872.
8. Begleiter R, Yaniv R. Superior guarantees for sequential prediction and lossless compression via alphabet decomposition. J Mach Learn Res. 2006;7:379–411.
9. Willems F, Tjalkens T. Complexity reduction of the context-tree weighting algorithm: a study for KPN research, EIDMA report RS.97.01. Eindhoven: Technical University of Eindhoven; 1997. https://www.researchgate.net/publication/228732029_Complexity_reducing_techniques_for_the_CTW_algorithm. https://core.ac.uk/display/56576627.
10. Tjalkens T, Willems F. Implementing the context-tree weighting method: arithmetic coding. In: International conference on combinatorics, information theory and statistics; 1997. p. 83.
11. Sadakane K, Okazaki T, Imai H. Implementing the context tree weighting method for text compression, Proceedings DCC 2000. Data Compression Conference, Snowbird, UT, 2000, pp. 123–32. https://doi.org/10.1109/DCC.2000.838152. https://dl.acm.org/citation.cfm?id=789787.
12. Tjalkens T, Volf P, Willems F. A context-tree weighting method for text generating sources. In: Data Compression Conference; 1997. p. 472.
13. Volf P. Weighting techniques in data compression theory and algorithms. Ph.D. thesis: Technische Universiteit Eindhoven; 2002. https://www.researchgate.net/publication/238123916_Weighting_techniques_in_data_compression_Theory_and_algorithms. https://www.scribd.com/document/63172312/Weighting-Techniques-in-Data-Compression-Theory-and-Algoritms.
14. Quddus MA, Noland RB, Ochieng WY. A high accuracy fuzzy logic based map matching algorithm for road transport. J Intell Transp Syst. 2006;10(3):103–15.
15. Greenfeld JS. Matching GPS observations to locations on a digital map. 81th annual meeting of the transportation research board. 2002. p. 164–73. https://www.researchgate.net/publication/246773761_Matching_GPS_Observations_to_Locations_on_a_Digital_Map.
16. Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters. In: Proceedings of the 6th conference on symposium on operating systems design & implementation. San Francisco; 2004. p. 10. https://dl.acm.org/citation.cfm?id=1327492.
17. Lammel R. Google's MapReduce programming model - revisited. Sci Comput Program. 2008;70:1–30.
18. Chang F, Dean J, Ghemawat S, Hsieh W, Wallach D, Burrows M, Chandra T, Fikes A, Gruber R. Bigtable: a distributed storage system for structured data. ACM Trans Comput Syst. 2008;26(2):1–26.
19. Haklay M, Weber P. OpenStreetMap: user-generated street maps. IEEE Pervasive Comput. 2008;7(4):12–8.
20. Ranade A. Mumbai Navigator. Indian J Transp Manag. 2005; https://www.cse.iitb.ac.in/~ranade/.

21. Rousell A, Hahmann S, Bakillah M, Mobasheri A. Extraction of landmarks from OpenStreetMap for use in navigational instructions. In: Proceedings of the AGILE conference on geographic information science. Lisbon; 2015. p. 9–12. https://agile-online.org/conference_paper/cds/agile_2015/posters/57/57_Paper_in_PDF.pdf. https://www.researchgate.net/publication/278301149_Extraction_of_landmarks_from_OpenStreetMap_for_use_in_navigational_instructions.
22. Zipf A, Mobasheri A, Rousell A, Hahmann S. Crowdsourcing for individual needs—The case of routing and navigation for mobility-impaired persons. In: Capineri C, Haklay M, Huang H, Antoniou V, Kettunen J, Ostermann F, Puves R, editors. European Handbook of Crowdsourced Geographic Information. London: Ubiquity Press; 2016. pp. 325–37.
23. Mobasheri A, Sun Y, Loos L, Ali AL. Are crowdsourced datasets suitable for specialized routing services? Case study of OpenStreetMap for routing of people with limited mobility. Sustainability. 2017;9(6):997.
24. Sun Y, Fan H, Bakillah M, Zipf A. Road-based travel recommendation using geo-tagged images. Comput Environ Urban Syst. 2015;53:110–22.
25. Bakillah M, Liang SHL, Mobasheri A and Zipf A. Towards an efficient routing web processing service through capturing real-time road conditions from big data, 2013 5th Computer Science and Electronic Engineering Conference (CEEC), Colchester, 2013, pp. 152–5. https://doi.org/10.1109/CEEC.2013.6659463.
26. Haworth B, Bruce E. A review of volunteered geographic information for disaster management. Geography Compass. 2015;9(5):237–50.
27. Zook M, Graham M, Shelton T, Gorman S. Volunteered geographic information and crowdsourcing disaster relief: a case study of the Haitian earthquake. World Med Health Policy. 2010;2(2):7–33.
28. Sun Y, Mobasheri A, Hu X, Wang W. Investigating impacts of environmental factors on the cycling behavior of bicycle-sharing users. Sustainability. 2017;9(6):1060.
29. Ganeshan K, Sarda L, Gupta S. Developing IITB smart CampusGIS grid. In: A2CWiC '10 Proceedings of the 1st Amrita ACM-W celebration on women in computing in India. New York: ACM; 2010.
30. Senaratne H, Mobasheri A, Ali AL, Capineri C, Haklay M. A review of volunteered geographic information quality assessment methods. Int J Geogr Inf Sci. 2017;31(1):139–67.
31. Mobasheri A, Huang H, Degrossi LC, Zipf A. Enrichment of OpenStreetMap data completeness with sidewalk geometries using data mining techniques. Sensors. 2018;18(2):509.
32. Mobasheri A. A rule-based spatial reasoning approach for OpenStreetMap data quality enrichment; case study of routing and navigation. Sensors. 2017;17(11):2498.
33. Aberg J, Shtarkov Y. Text compression by context tree weighting. In: Proceedings data compression conference (DCC); 1997. p. 377–86.
34. Willems F. The context-tree weighting method: extensions. IEEE Trans Inf Theory. 1998;44(2):792–8.
35. Willems F, Shtarkov Y, Tjalling T. Context weighting for general finite-context sources. IEEE Trans Inf Theory. 1996;42(5):1514–20.
36. Willems F. Coding for a binary independent piecewise-identically-distributed source. IEEE Trans Inf Theory. 1996;42(11):2210–7.
37. Simmons R, Browing B, Yilu Z, Sadekar V. Learning to predict driver route and destination intent. In: Intelligent transportation systems conference; 2006.
38. Burbey I, Martin TL. Predicting future locations using prediction-by-partial-match. In: Proc. 1st ACM MELT; 2008. p. 1–6.
39. Tiwari VS, Chaturvedi S, Arya A. Route prediction using trip observations and map matching, 2013 3rd IEEE International Advance Computing Conference (IACC), Ghaziabad, 2013, pp. 583–7. https://doi.org/10.1109/IAdCC.2013.6514292.
40. Lung HY, Chung CH, Dai B-R. Predicting locations of mobile users based on behavior semantic mining. In: Trends and applications in knowledge discovery and data mining, lecture notes in computer science, vol. 8643; 2014.
41. Neto FDN, Baptista CDS, Campelo CEC. Prediction of destinations and routes in urban trips with automated identification of place types and stay points. In: Proc. Brazilian Symposium on Geoinformatics; 2015. p. 80–91.
42. Amirat H, Lagraa N, Fournier Viger P, Ouinten Y. MyRoute: a graph-dependency based model for real-time route prediction. J Commun. 2017; https://doi.org/10.12720/jcm.12.12.668-676.
43. Tiwari VS, Arya A. Horizontally scalable probabilistic generalized suffix tree (PGST) based route prediction using map data and GPS traces. Journal of Big Data. 2017;4:23. https://journalofbigdata.springeropen.com/articles/10.1186/s40537-017-0085-4.