บทความวิจัย (Research Article)

Modeling HMM Map Matching Using Multi-label Classification

Atichart Sinsongsuk, Thanapa Boonchoo and Wanida Putthividhya*

Department of Computer Science, Thammasat University (Rangsit Campus)
* Corresponding author
wanidap@cs.tu.ac.th

Accepted: 31 Dec 2021

Abstract

Map matching deals with matching GPS coordinates to corresponding points or segments on a road network map. The work has various applications in both vehicle navigating and tracking domains. Traditional rule-based approach for solving the Map matching problem yielded great matching results. However, its performance depends on the underlying algorithm and Mathematical/Statistical models employed in the approach. For example, HMM Map Matching yielded $O(N^2)$ time complexity, where N is the number of states in the underlying Hidden Markov Model. Map matching techniques with large order of time complexity are impractical for providing services, especially within time-sensitive applications. This is due to their slow responsiveness and the critical amount of computing power required to obtain the results. This paper proposed a novel data-driven approach for projecting GPS trajectory onto a road network. We constructed a supervised-learning classifier using the Multi-Label Classification (MLC) technique and HMM Map Matching results. Analytically, our approach yields O(N) time complexity, suggesting that the approach has a better running performance when applied to the Map matching-based applications in which the response time is the major concern. In addition, our experimental results indicated that we could achieve Jaccard Similarity index of 0.30 and Overlap Coefficient of 0.70.

Keywords: Map matching, Hidden Markov Model, Multi-Label Classification, Supervised-learning, Machine learning

1. Introduction

Map matching enhances the benefit of GPS by projecting GPS latitude/longitude pairs measured over time onto a logical model of the real world, for example, a road network map. Being able to locate GPS observations of objects on a logical road network map is beneficial and has applications in both vehicle navigating and tracking domains - pedestrian and vehicle navigation systems, freight management used to draw routes visited by company vehicles, and the "pay as you go" services used to collect toll fee to name a few.

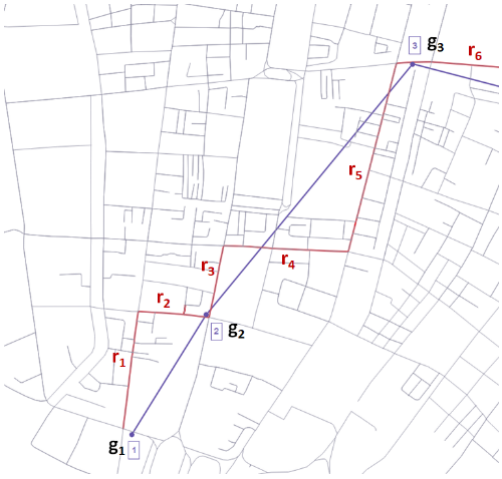


Fig.1 Main components of the Map Matching problem consists of measured GPS coordinates (numbered black dots), corresponding GPS trajectories (purple lines), and the actual routes traversed by a vehicle (connected red lines).

Fig. 1 demonstrates the input and the desired output of the Map Matching problem. The sequence of GPS coordinates measured over time by a moving object is the input of the problem. The figure also displays GPS trajectories corresponding to the measured GPS coordinates. The goal is to discover a sequence of road segments on a given road network map the moving object actually traversed.

Solving the Map Matching problem is quite challenging. Difficulties are caused by the intrinsic characteristic of satellite signal, complex city plans, and not well-organized urban road 15 to more than 50 meters [4]. The error could be worse in an urban setting compared to when GPS receivers have wide visibility from satellites. This is due to signal bouncing off of obstacles, for example, buildings, narrow streets, trees, sky trains, or cars. As a result, no GPS coordinates can be computed and recorded into a data set for a period of time. Being aware of periods with missing GPS observations in the data set is critical. Appropriate actions are required to prepare data prior to feeding the data set into any algorithm or model for solving the Map Matching problem.

Applying data-driven [5-6] approach for solving the Map Matching problem deserves more attention. With the data-driven approach, solutions for solving the Map Matching problem work in two phases, namely the offline training phase and the online phase. During the offline training phase, they employed different machine learning (ML) or deep learning (DL) algorithms to learn relationship between GPS observations and vectors of road segments. Afterwards, machine learning or deep learning models were constructed and tuned accordingly. In an online phase, the models could be used to map unseen GPS trajectories to vectors of road segments.

Addressing the Map Matching problem using such data-driven approach could be beneficial. This is due to its small order of temporal complexity. That is, despite the time taken to train the model in an offline phase, the model could provide a vector of road segments that match given GPS trajectories within a very small amount of time during the online phase. Time complexity for finding a solution depends on the number of features to be considered when constructing the models.

In this work, we proposed a data-driven approach for solving the Map Matching problem. We defined the Map Matching problem as a supervised-learning multi-label classification task. The results yielded by the HMM Map Matching algorithm [1] were used as ground truth for constructing our models. The results of HMM Map Matching algorithm usually yield a 1-to-many mapping between a GPS coordinate and road segments. This fact suggested the use of multi-label classification

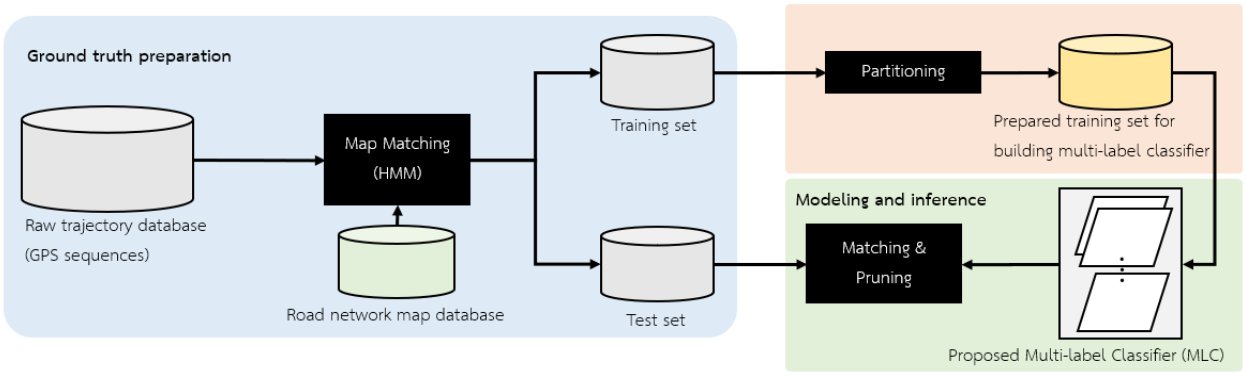


Fig. 2 The overall proposed method.

techniques for constructing our models. We constructed multi-label classification (MLC) models based on One-vs-Rest (OvR) [8] and One-Class Classification (OCC) [9] algorithms. During the offline learning phase, our models learned the mappings between GPS trajectories and road segments yielded by the HMM Map Matching algorithm as correct answers. Then, given an unseen GPS trajectory in the online phase, it is desirable that road segments resulting from our models were similar to those returned from the HMM Map Matching algorithm. The performance of our MLC models were evaluated using two set similarity indices [10], namely Jaccard Similarity index [11-12] and Overlap Coefficient [13]. Both metrics evaluate the model's performance in terms of the similarity between sets resulting from our models and the ground truth. The closer the metric is to 1.0, the higher the model performance. Employing both performance metrics in our experiments helped to identifying the direction for improving the performance of our MLC-OCC models.

The rest of this paper is organized as follows. Section 2 discussed briefly existing work relating to the Map Matching problem. Next, the work modeling HMM Map Matching using multi-label classification was presented in Section 3. Then, we presented and discussed our experimental results in Section 4. Lastly, we drew our conclusion and future work in Section 5.

2. Related Work

Related work could be divided into two categories. The first group [1-3] consists of those employing the rule-based approach for solving the Map Matching problem. On the other hand, the second group [5-6] consists of those using the data-driven approach for improving the positional accuracy of observed GPS coordinates. The accuracy of observed GPS coordinates has a big impact on the performance of map matching algorithms.

Hidden Markov Model (HMM) Map Matching [1] proposed a novel and principled map matching algorithm based on a Hidden Markov Model and the Viterbi algorithm. The algorithm seeks for the most likely road route corresponding to a sequence of timestamped GPS observations. HMM Map Matching yielded $O(N^2)$ time complexity where N is the number of states in the underlying Hidden Markov Model [1]. An improved HMM-based method, Fast map matching (FMM) [2], improved the performance of the original HMM method [1] by proposing an upper bounded origin-destination table to store all pairs of the shortest paths within a certain length in the road network. Repeated routing queries known as the bottleneck of

HMM [1] are then replaced by hash table search. As a result, time required to complete map matching could be reduced by 10 seconds. However, FMM may still suffer from constructing such precomputed table when the database contains the excessive number of road segments. While [1-2] were offline map matching algorithms, another work [3] proposed an online map matching algorithm based on the second-order Hidden Markov model (HMM) with a number of additional factors such as drivers' travel preference, road level, and vehicle heading. An extended Viterbi algorithm and a self-adaptive sliding window mechanism were adopted. The work found that the second-order HMM method outperformed the baseline algorithm built on the first-order HMM in various testing experiments.

Recently, there were a few works employing machine learning (ML) algorithms on the Map matching problems. The work proposed by [5] uses a well-known ML algorithm, i.e., multilayer perceptron (MLP), to regress the next GPS coordinate based on the horizontal GPS delusion and vehicles' speed, thereby improving the accuracy of originally observed GPS coordinates. However, the authors found that such method could not improve the accuracy of map matching algorithm. An extended work [6] employed more ML algorithms, i.e., Least Squares (LS), Multi-Layer Perceptron (MLP), and K-Nearest Neighbors (KNN), to instead predict GPS delusion based on location and time. The model with KNN helped double the positional accuracy of unseen GPS points. Once the higher positional accuracy can be yielded, the performance of map matching algorithm can consequently be improved.

Our work is different from existing work relating to the Map Matching problem in the following aspects. While the existing work [1-3] employed the rule-based approach for addressing the Map Matching problem, our work implemented the data-driven approach to solve the problem. Even though the work [5-6] also employed the data-driven approach, their goals are different from ours, attempting to model the result of HMM Map Matching [1] via supervised-learning, multi-label classification.

3. Methodology

In this section, we detail our proposed model. The overall model is shown in Fig. 2. Our proposed model consists of three main processes: (1) Map matching using HMM, (2) Partitioning, and (3) Pruning, which can be referred to as black boxes in the Fig. 2. In what follows, we will describe each process separately.

3.1 Map Matching using HMM

A collected trajectory database contains M trajectories where a trajectory T^i is defined as a sequence of GPS $T^i = \left\{\left(x_1^i, y_1^i\right), \left(x_2^i, y_2^i\right), \cdots, \left(x_{|T^i|}^i, y_{|T^i|}^i\right)\right\}, 1 \leq i \leq M$, and $|\mathrm{T}^{\mathrm{i}}|$ is the length of trajectory T^i. This trajectory database can be collected by a GPS receiver residing inside a GPS-quipped on-road moving objects, e.g. taxi, that measures its current GPS coordinate. However, these measured GPS coordinates can be noisy due to the aforementioned causes (see Section 1). Therefore, Map matching algorithm plays an important role to map such unreliable GPS coordinates to correct road segments on road network map. Specifically, given a trajectory T , the Map matching algorithm finds the most reasonable *path* with which the trajectory T^i will match, where the *path* is defined as a sequence of road segments $P^i = \{r_1^i, r_2^i, \cdots, r_{|P^i|}^i\}$ on a road network map database, where $|P^i|$ is the length of the path P^i. It is worth noting that although the path P^i corresponds to

the trajectory T^i, the lengths of both path P^i and trajectory T^i can be different.

The HMM Map Matching algorithm [1-2] takes the full connectivity of all road segments into consideration regarding their transition probabilities to perform matching, the algorithm therefore can produce reliable result which has the highest probability among the others. The HMM Map Matching algorithm can produce a promising result and has been proven to be effective on various map-based applications. Thus, we decided to use it to perform the matching. The HMM algorithm regards GPS coordinates and road segments as observations and hidden states, respectively, Consider Fig. 1 as an example. The trajectory T^a (numbered black dots) containing three GPS coordinates $T^a = \{g_1, g_2, g_3\}$ matches to the underlining path $P^a = \{r_1, r_2, r_3, r_4, r_5, r_6\}$, and $|T^a| \neq |P^a|$.

3.2 Partitioning

After HMM has been constructed, we can use the constructed HMM to transform the raw trajectory database containing M trajectories $T = \{T^1, T^2, \cdots, T^M\}$ to its road path database $P = \{P^1, P^2, \cdots, P\text{^}M\}$, where path P^i is the output from HMM that accepts the trajectory T^i as the input. In other words, we can view HMM as a mapping function $f: T \rightarrow P$.

Here, the partitioning process is to make the training set appropriate to feed into the multi-label classifier model (MLC) in which we will give the details of MLC in the next subsection. In this perspective, we define the Map Matching problem as a supervised learning classification problem. As mentioned in the previous subsection, the length of given trajectory T^i may not be the same as the length of its corresponding HMM path P^i. Consequently, we decided to use the multi-label classification for such supervised learning problem. Therefore, we propose a partitioning method to find proper alignments between the GPS coordinates and the output road segments. Specifically, one GPS coordinate (x, y) is treated as the set of features for MLC to learn the mapping with the target variables (road segments).

In this partitioning process, apart from a trajectory $T^i = \{(x_1^i, y_1^i), (x_2^i, y_2^i), \cdots, (x_{|T^i|}^i, y_{|T^i|}^i)\}$ and its corresponding path $P^i = \{r_1^i, r_2^i, \cdots, r_{|P^i|}^i\}$, we also have a set of pivots $\Pi^i = \{\rho_1^i, \rho_2^i, \cdots, \rho_{|T^i|}^i\}$, where $1 \leq \rho_j^i \leq |P^i|$ and $|T^i| = |\Pi^i|$, such that GPS coordinate (x_j^i, y_j^i) and $r_{\rho_j^i}^i$ are ideally the 1-1 best alignment[1]. Thus, given P^i and Π^i, the partitioning process finds the alignment for any given GPS coordinate $(x_j^i, y_j^i) \in T^i$ as $\alpha_j^i = (x_j^i, y_j^i) \rightarrow \{r_{\rho_j^i - w}^i, \cdots, r_{\rho_j^i - 1}^i, r_{\rho_j^i}^i, r_{\rho_j^i + 1}^i, \cdots, r_{\rho_j^i + w}^i\}$. Here we have an additional parameter w in the model. Since w should be carefully selected and the lengths of trajectories in T and the lengths of their corresponding path in P are different, we decide to use an adaptive strategy to set the w depending on the length of the considered trajectory and its corresponding path. That is, given that we are considering trajectory T and path P, w can be calculated by $w = Ceil(\frac{|P|}{|T|})$.[2] Then, given a trajectory $T^i = \{(x_1^i, y_1^i), (x_2^i, y_2^i), \cdots, (x_{|T^i|}^i, y_{|T^i|}^i)\}$, the partitioning process will return an alignment set $A^i = \{\alpha_1^i, \alpha_2^i, \cdots, \alpha_{|T^i|}^i\}$ which can facilitate the next process to train the model.

3.3 Multi-label Classifier Model building

In this subsection, we describe how the multi-label classifier model (MLC) is built. The main task of MLC is

[1] An 1-1 best alignment can be guided by the process of Map matching using HMM

[2] Notice that if we statically set w as 0, it means an extreme case that we only align each GPS coordinate with its corresponding pivot.

to learn the mapping between a trajectory and a path produced by HMM Map Matching which is a rule-based method that we have already gone through from the Section 3.1. From the partitioning method in Section 3.2, we can obtain the alignment set of all pairs of trajectories and paths which are the output of the HMM Map Matching method (refer to Section 3.1). Thus, given a trajectory database $T = \{T^1, T^2, \cdots, T^M\}$, we can also obtain the corresponding alignment database $A = \{A^1, A^2, \cdots, A^M\}$, where $A^i = \{\alpha_1^i, \alpha_2^i, \cdots, \alpha_{|T^i|}^i\}$ and $\alpha_j^i = [(x_j^i, y_j^i) \rightarrow \{r_{\rho_{j-w}^i}^i, \cdots, r_{\rho_{j-1}^i}^i, r_{\rho_j^i}^i, r_{\rho_{j+1}^i}^i, \cdots, r_{\rho_{j+w}^i}^i\}]$.

Considering the alignment mapping α_j^i, we decide to model this mapping using a multi-label classifier. More specifically, in practice such mapping can be very skewed, i.e. it is possible that a road segment can have only few GPS mapped on it, we therefore choose One-Class classification algorithm (MLC-OCC) as the implementation of the multi-label classifier instead of One-vs-Rest (MLC-OvR) as we will show in the experiment section that MLC-OvR will suffer from the skewness of the training data.

One-Class classification (OCC) is a method that learns a representation of a classifier that is capable of recognizing positive labeled instances during the prediction. Since one GPS coordinate can map with more than one road segment, we can construct a multi-label classifier by regarding one OCC model for one road segment. Therefore, the proposed multi-label classification model will consist of a set of OCC classifiers, i.e. $C = \{C_{r_1}, C_{r_2}, \cdots, C_{r_N}\}$, where N is the number of road segment in the road network map database. We can regard a OCC classifier C_{r_i} as a function from GPS coordinates (x, y) to $\{+, -\}$, where $+$ denotes that the GPS (x, y) corresponds to road segment r_i, while $-$ denotes otherwise. In other words, the function C_{r_i} can be written as a function $C_{r_i}: (x, y) \rightarrow \{+, -\}_{r_i}$.

Consequently, given a GPS coordinate, the multi-label classifier will undergo a test for all OCC classifiers in C, and thus yield a set of correspondences. Note that the function C_{r_i} can be implemented using any OCC algorithms [8].

3.4 Matching and Pruning

Once MLC $C = \{C_{r_1}, C_{r_2}, \cdots, C_{r_N}\}$ has been built, we can perform inference on given GPS coordinates. However, both GPS coordinates and road network map data have intrinsically a spatial characteristic, we additionally introduce two new attributes of a road segment r which are $r.x$ and $r.y$ representing position in the same space as the GPS coordinates. The procedure for matching GPS to a set of road segments is presented in Algorithm 1.

Algorithm 1 Matching and Pruning

```
1: procedure MATCH(C, R, p, K)
2:     RS ⟵ ∅
3:     R_pruned ⟵ PRUNE(R, p, s)
4:     for r ∈ R_pruned do
5:         if C_r(p) belongs to + then
6:             RS.insert(r)
7:         end if
8:     end for
9:     Return RS
10: end procedure
11: procedure PRUNE(R, p, K)
12:     R_pruned ⟵ NearestNeighbours(p, K)
13:     Return R_pruned
14: end procedure
```

No	Observed GPS Trajectory	Sequence of Corresponding Road Segments: pivot (opath) and corresponding path(cpath) produced by HMM
1	(-8.609067, 41.144427) , (-8.609058, 41.144427) , (-8.609049, 41.144409) , (-8.609049, 41.144364) , (-8.60904, 41.144373)	opath : 119 , 119 , 119 , 119 , 119 cpath : 119
2	(-8.614125, 41.147703) , (-8.613837, 41.147631) , (-8.613702, 41.147604) , (-8.613693, 41.147604)	opath : 107 , 159 , 46 , 46 cpath : 107 , 159 , 46
3	(-8.61264, 41.146047) , (-8.61219, 41.14602) , (-8.612271, 41.147163) , (-8.612469, 41.147892) , (-8.621703, 41.147541) , (-8.62173, 41.147487) , (-8.621739, 41.147478) , (-8.62173, 41.147469)	opath : 30 , 11 , 11 , 87 , 13 , 13 , 13 , 13 cpath : 30 , 11 , 6 , 8 , 90 , 87 , 157 , 89 , 52 , 14 , 13

Fig. 3 Example of results yielded by HMM: observed GPS trajectory, pivot (opath) and corresponding path representing by road segment ID (cpath)

The procedure MATCH accepts four parameters: the trained MLC C, the road network map database R, the given GPS coordinate p and the pruning parameter d. The procedure MATCH must undergo a test for all road segments for correctness; however, we argue that some road segments are unnecessary to consider if their distances from p are too long. For example, if coordinate p is 100 kilometers far from a road segment r_f, so it is impossible that point p will match with r_f by HMM at the first place.

Thus, we further propose the procedure PRUNE to prune such impossible road segments so that it will return only the set of road segments that possibly matches with the points p to perform the matching procedure in the procedure MATCH as shown in Algorithm 1. In practice, the number of road segments in R can be tremendous so that it can impact the overall running performance or even accuracy performance of the procedure MATCH. The procedure PRUNE can prune the unnecessary road segments to under the test in MATCH. We argue that there are many ways to implement the procedure to PRUNE. In this paper, we choose the well-known K nearest neighbor (KNN for short) algorithm since KNN can pick first K road segments that are nearest to the coordinate p and it discards other road segments whose distances would possibly be far from p.

In the process of inference, after pruning we only need to perform the classification task on only K classifiers where $1 \leq K \leq N$. Let c is the number of learned parameters in each classifier. Therefore, the process of inference takes $cK \approx O(N)$ time to run for each mapping between a given GPS coordinate and matched road segments.

4. Experimental and Results

In this section, first, we describe the data set utilized in our experiments and our experimental setup. Then, we report the experimental results produced by our proposed MLC-OCC models. Our experiments focus on measuring the performance of our models by computing similarity between the answer sets resulting from our models and the ground truth produced by HMM.

4.1 Data Set

In this paper, we utilized part of the Porto taxicabs' GPS coordinates data set [7]. The data set consists of a sequence of GPS trajectories measured over time by 442 taxicabs serving around Porto metropolitan area, Portugal. Each GPS trajectory, in turn, consists of a

Table 1 MLC-OCC Configuration settings for evaluating performance of Multi-Label Classification using One-Class Classifier (MLC-OCC) models

No.	Configuration Settings	Explanations
1	Partitioning (w=0)	Corresponding paths produced by HMM were partitioned based on window size equal to zero.
2	Partitioning (Adaptive)	Corresponding paths produced by HMM were partitioned based on the adaptive window size.
3	Partitioning (w=0) + Pruned (K=3)	Corresponding paths produced by HMM were partitioned based on window size equal to zero, and 3 road segments were selected to perform the matching procedure.
4	Partitioning (Adaptive) + Pruned (K=n)	Corresponding paths produced by HMM were partitioned based on the adaptive window size and a number of road segments were selected to perform the matching procedure, where K was a number of road segments were selected. In this work was used K equal to 3, 5, 7 and 15.

sequence of GPS coordinates measured over time. Each GPS trajectory represents a route traversed by participating GPS-equipped taxicabs. Our goal is to have our MLC-OCC models correctly project this route onto the road network map.

Currently, our data set contains 20,939 GPS trajectories and 229 road segments. Among these, 16,751 GPS trajectories (189,348 GPS points) were used as our training set during to construct our MLC-OCC models during the training phase. In addition, 1,477 GPS trajectories (48,049 GPS points) resided in our test set to be used during the testing phase of our experiments.

4.2 Data Preparation

Initially, we deployed the HMM Map Matching algorithm made available via the HMM Map Matching framework [1-2] to prepare the ground truth data for modeling HMM using supervised-learning multi-label classification technique. The framework accepts a Sequence of GPS trajectories measured over time and a road network map as its inputs. Then, it yields a matching between each GPS trajectory and corresponding sequence of road segments on the road network map. The sequences of road segments resulting from HMM, when drawn on the road network map, represent routes traversed by vehicles. Fig. 3 demonstrates an example of results yielded by the HMM framework. For each GPS trajectory, the framework provides a sequence of corresponding road segments on the road network map (cpath) and the list of pivots which are road segments ideally 1-1 best aligning with GPS points (opath). Fig. 4 shows an example of road segments corresponding to each GPS point after partitioning based on adaptive window size (w) and the resulting training set for constructing each MLC-OCC classifier. Recall that the number of road segments determines the number of classifiers in our proposed work.

4.3 Performance Metrics

Given an unseen GPS trajectory, the effectiveness of our proposed MLC-OCC models could be evaluated by measuring the similarity between two sets, i.e. the set of road segments returned from our MLC-OCC models and the results yielded by HMM when attempting to project the given GPS trajectory onto a road network map.

We employed two performance metrics which are commonly used for measuring set similarity, namely Jaccard similarity index (JAC) and Overlap Coefficient (OLC). The JAC and OLC performance metrics can be defined as follows.

$$JAC(M,G) = \frac{|M \cap G|}{|M \cup G|} = \frac{|M \cap G|}{|M| + |G| - |M \cap G|} \quad (1)$$

$$OLC(M,G) = \frac{|M \cap G|}{\min(|M|,|G|)} \quad (2)$$

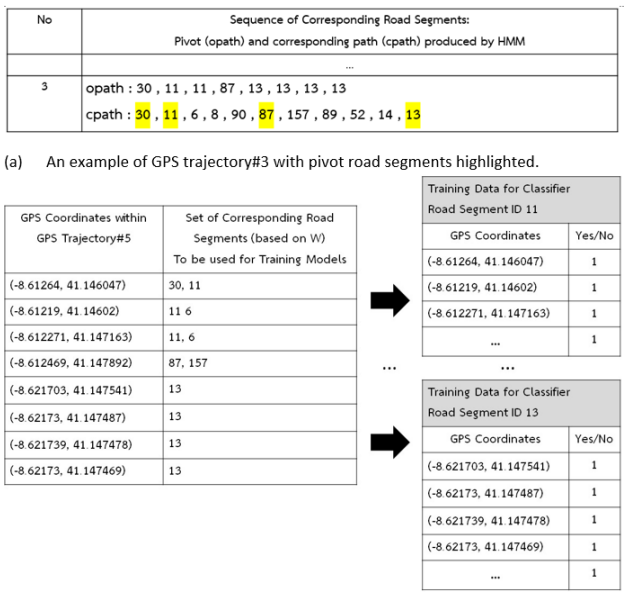
Where M and G are sets resulting from our MLC-OCC models and our ground truth HMM Map Matching, respectively.

Both JAC and OLC performance metrics fall within the range from 0.0 to 1.0. The closer the metric is to 1.0, the more the two sets are similar.

JAC captures the performance of our MLC-OCC models in various scenarios. The numerator in JAC's definition $(|M \cap G|)$ quantifies the common part of the two sets M and G. On the other hand, the denominator part in JAC's definition $(|M \cup G|)$ covers both common and different parts of the two sets. Henceforth, the denominator part in JAC's definition reflects how much our models could classify road segments correctly and how much our models do in a wrong manner. The wrong manner could be divided further into two possibilities, i.e. road segments appearing in M but not in G (false positive) and those appearing in G but not in M (false negative).

Unlike JAC, OLC focuses more on the true positive scenario by accounting for road segments appearing in both sets M and G in the numerator part of its definition but employing the cardinality of the smaller set as the denominator. Compared to JAC, the denominator in OLC's definition lessen the impact of the false positive and false negative manners done by our MLC-OCC models.

Employing both performance metrics in our experiments helped to identifying the direction for improving the performance of our MLC-OCC models.



No	Sequence of Corresponding Road Segments: Pivot (opath) and corresponding path (cpath) produced by HMM
	...
3	opath : 30 , 11 , 11 , 87 , 13 , 13 , 13 , 13 cpath : 30 , 11 , 6 , 8 , 90 , 87 , 157 , 89 , 52 , 14 , 13

(a) An example of GPS trajectory#3 with pivot road segments highlighted.

GPS Coordinates within GPS Trajectory#5	Set of Corresponding Road Segments (based on W) To be used for Training Models
(-8.61264, 41.146047)	30, 11
(-8.61219, 41.14602)	11 6
(-8.612271, 41.147163)	11, 6
(-8.612469, 41.147892)	87, 157
(-8.621703, 41.147541)	13
(-8.62173, 41.147487)	13
(-8.621739, 41.147478)	13
(-8.62173, 41.147469)	13

Training Data for Classifier Road Segment ID 11

GPS Coordinates	Yes/No
(-8.61264, 41.146047)	1
(-8.61219, 41.14602)	1
(-8.612271, 41.147163)	1
...	1

...

Training Data for Classifier Road Segment ID 13

GPS Coordinates	Yes/No
(-8.621703, 41.147541)	1
(-8.62173, 41.147487)	1
(-8.621739, 41.147478)	1
(-8.62173, 41.147469)	1
...	1

Fig. 4 An example of HMM result, partitioned road segments, and training data for constructing MLC-OCC classifiers.

4.4 Experimental Setup

We evaluated the performance of our proposed MLC-OCC models based on seven different configuration setting as listed in Table 1. Each configuration setting differs from the others in two following aspects. The first aspect is how corresponding paths resulting from HMM is handled in the partitioning phase (Section 3.2). And, the second aspect is the number of road segments (K) resulting from the procedure PRUNE to perform the matching procedure (Section 3.4). The performance of our proposed models in different settings described in Table 1 would be beneficial for briefly drawing an impact of adaptive window size and an impact of the procedure PRUNE to the performance.

4.4.1 Testbed

We conducted all the experiments on a computer equipped with Intel Core i7-7700 @ 3.60GHz 16 GB RAM running 16.40 Ubuntu operating system. We used the open-source map matching framework available

online[3]. All the models and processes proposed in this paper were implemented based on Scikit-learn and Python 3.9.

Table 2 Accuracy performance of MLC-OCC models

No.	MLC-OCC Configuration Settings / Similarity Metrics	JAC	OLC
1	Partitioning (w=0)	0.03	0.36
2	Partitioning (Adaptive)	0.05	0.7
3	Partitioning (w=0) + Pruned (K=3)	0.1	0.24
4	Partitioning (Adaptive) + Pruned (K=15)	0.26	0.67
5	Partitioning (Adaptive) + Pruned (K=7)	0.28	0.65
6	Partitioning (Adaptive) + Pruned (K=5)	0.29	0.61
7	Partitioning (Adaptive) + Pruned (K=3)	0.30	0.53

4.5 Accuracy Performance

Table 2 demonstrates our experimental results. Each row presents the performance of our MLC-OCC models in a specified configuration setting. The performance of our models was evaluated in terms of the Jaccard Similarity index (JAC) and the Overlap Coefficient (OLC) as stated earlier in Section 4.3.

In the first configuration setting where adaptive window of size 0 was applied when partitioning paths resulting from HMM, our model achieved JAC of 0.03 and OLC of 0.36. However, when applying adaptive window size w to partition the HMM's results, we achieved a slightly better JAC (i.e. 0.04) and a twofold increase in OLC (i.e. 0.70). Recall that the current version of our work uses an adaptive strategy to set the window size w depending on the length of the considered GPS trajectory and its corresponding path. Results from these first two configuration settings imply the accuracy performance of our MLC-OCC models. To be more specific, the twofold increase in only OLC implies that the adaptive window size strategy helps increase the number of road segments in set M which are in common with those in our ground truth set G.

Next, the third configuration setting evaluate the performance of our models, with some unrelated road segments being pruned prior to the matching procedure. The adaption window size is set to zero in order to merely see an impact of the PRUNE procedure. Experimental results revealed that our models

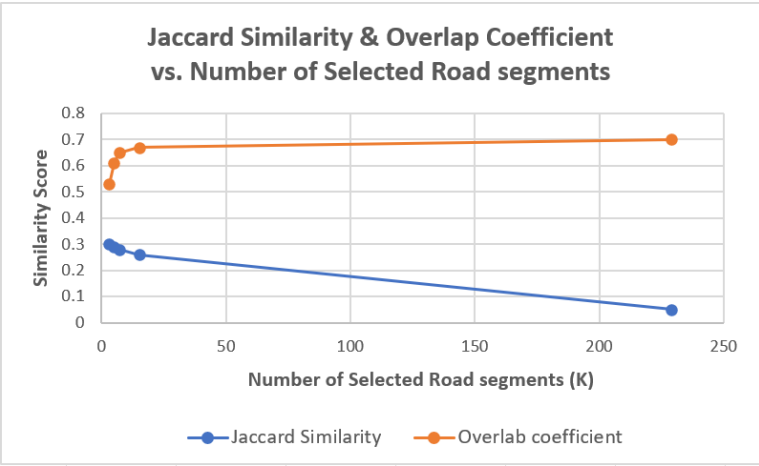


Fig. 5 Jaccard Similarity and Overlap Coefficient result vs. Number of selected road segments

[3] https://fmm-wiki.github.io/

achieved JAC of 0.1 and OLC of 0.24. When compared to the model's performance in the first configuration setting, JAC in the third setting got a threefold increase, while OLC was slightly decreased. Results from the first and third configuration settings imply that the PRUNE procedure could improve the accuracy performance of our MLC-OCC models as well. From Fig. 5, we noticed that as the number of selected road segments (K) decreases, JAC increases but OLC decreases. With the adaptive strategy for computing a sliding window and the use of K-Nearest Neighbors in selecting the number of road segments, the highest JAC our models could achieve is 0.30 with the corresponding OLC equal to 0.53.

4.6 Discussion

As suggested by the experiment, we found that the processes of partitioning and pruning have a significant impact on the model accuracy performance (see Section 4.5). For the partitioning process, we introduced an adaptive strategy with regard to the trajectory length to perform the alignment with the pivots, and the ablation test showed the notable improvement when the method is utilized. This finding also demonstrates that the alignment strategy clearly affects the accuracy performance for both metrics.

Likewise, the pruning method was devised to help the model improve in two aspects: accuracy and running performance in the stage of inference. As intended, the pruning method chose only the first K necessary classifiers to perform inference while discarding those unnecessary. As a result, the model with the pruning method achieved a better JAC. The reason is that the pruning method reduces $false\ positive$ of the model because it will not consider the classifiers of road segments that are too far from the given GPS coordinate in the stage of inference.

In this paper, although considering different algorithms for the processes of partitioning and pruning is beyond the scope of this paper, this still makes room for future research to explore more strategies for the two processes. For example, instead of the K nearest neighbors as the pruning method, we may adopt a method that prunes classifiers whose distances to the given GPS coordinate is longer than a pre-defined threshold. This might result in improving those metrics.

5. Conclusion and Future Work

This paper introduced a novel data-driven approach for solving the Map matching problem. We viewed the Map matching problem as a model of well-developed Hidden Markov Model-based Map Matching algorithm (HMM Map matching) learnt by a supervised-learning multi-label classification method. We found that using One-vs-Rest (MLC-OvR) algorithm may not be appropriate to build the model due to the problems of skewness and imbalance characteristics of the data. In this paper, we proposed to adopt the One-Class Classification (MLC-OCC) algorithm to build the model and found that the proposed MLC-OCC achieved up to 0.30 and 0.70 for Jaccard Similarity index and Overlap Coefficient which were far better than that of MLC-OvR.

In particular, in the proposed MLC-OCC framework, we proposed two additional processes that contributed to the accuracy performance of the model. First, we proposed an adaptive partitioning process to dynamically align a GPS coordinate with a set of road segments. Second, we devised the pruning method such that the model can discard some unnecessary classifiers to consider in the stage of inference. As

shown in the experiments, both partitioning and pruning process improved the accuracy performance on both Jaccard Similarity index and Overlap Coefficient significantly.

For future work, since our proposed model produced the output as a set of possible road segments that are likely to match the given input trajectory, it would be interesting to consider the connectivity information in the road network map database to help the alignment. Another challenge would be to extend this work to other databases of trajectories and road networks to observe the behaviors of the output models on different datasets. Moreover, the proposed model could also be interesting when employing the other ground-truth Map matching algorithms and examine the resulting models.

6. References

[1] P. Newson and J. Krumm, "Hidden markov map matching through noise and sparseness," in Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems, pp. 336–343, 2009.

[2] C. Yang and G. Gidofalvi, "Fast map matching, an algorithm integrating hidden markov model with precomputation," International Journal of Geographical Information Science, pp. 1–24, 2017.

[3] X. Fu, J. Zhang, and Y. Zhang, "An online map matching algorithm based on second-order hidden markov model," Journal of Advanced Transportation, vol. 2021, p. 9993860, 2021.

[4] Carlos III University of Madrid, (16 December 2021), Precision of GPS in cities improved by 90 percent [Online] Available: https://www.sciencedaily.com/releases/2013/02/130212121858.htm

[5] M. Hashemi and H. A. Karimi, "A machine learning approach to improve the accuracy of gps-based mapmatching algorithms (invited paper)," in 2016 IEEE 17th International Conference on Information Reuse and Integration (IRI), pp. 77–86, 2016.

[6] M. Hashemi, "Reusability of the output of map-matching algorithms across space and time through machine learning," IEEE Transactions on Intelligent Transportation Systems, vol. 18, no. 11, pp. 3017–3026, 2017.

[7] D. Dua and C. Graff. (4 December 2021). UCI machine learning repository: Taxi service trajectory - prediction challenge, ecml pkdd 2015 data set, 2017. [Online] Available: http://archive.ics.uci.edu/ml

[8] J. Brownlee. (4 December 2021), One-vs-rest and one-vs-one for multi-class classification. [Online] Available: https://machinelearningmastery.com/one-vsrest-and-one-vs-one-for-multi-class-classification/

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, (4 Deccember 2021) Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011. [Online] Available: https://scikitlearn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html

[10] M. K. Vijaymeena and K. Kavitha, "A survey on similarity measures in text mining," (PDF). Machine Learning and Applications. 3, vol. 3, pp. 19-28, 2016.

[11] Neo4j, (4 December 2021), The World's Leading Graph Database, Std.,. [Online] Available: http://neo4j.org/

[12] Wikipedia. (16 December 2021), Jaccard index — Wikipedia, the free encyclopedia [Online] Available: http://en.wikipedia.org/w/index.php?title=Jaccard%20index&oldid=1040991564

[13] Wikipedia. (16 December 2021), Overlap coefficient — Wikipedia, the free encyclopedia [Online] Available: http://en.wikipedia.org/w/index.php?title=Overlap%20coefficient&oldid=102201613