Grand Challenge: MtDetector: A High-performance Marine Traffic Detector at Stream Scale

Chun-Xun Lin*
ECE Dept, UIUC, IL
clin99@illinois.edu

Tsung-Wei Huang*
ECE Dept, UIUC, IL
twh760812@gmail.com

Guannan Guo
ECE Dept, UIUC, IL
guannan4@gmail.com

Martin D. F. Wong
ECE Dept, UIUC, IL
mdfwong@illinois.edu

ABSTRACT

In this paper, we present *MtDetector*, a high performance marine traffic detector that can predict the destination and the arrival time of travelling vessels. MtDetector accepts streaming data reported by the moving vessels and generates continuous predictions of the arrival port and arrival time for those vessels. To predict the destination for a ship, MtDetector builds a neural network for every port and infers the arrival port for vessels based on their departure port. For the arrival time prediction, we derive informative features from training data and apply Deep Neural Network (DNN) to estimate the traveling time. MtDetector is built on top of *DtCraft* [1, 2], a high-performance distributed execution engine for stream programming. By utilizing the task-based parallelism in DtCraft, MtDetector can process multiple predictions concurrently to achieve high throughput and low latency.

CCS CONCEPTS

• **Theory of computation → Distributed computing models**; • **Computing methodologies → Neural networks**; • **Software and its engineering** → *Cloud computing*;

KEYWORDS

Distributed System, Marine Traffic, Machine Learning, Stream Processing

ACM Reference Format:
Chun-Xun Lin, Tsung-Wei Huang, Guannan Guo, and Martin D. F. Wong. 2018. Grand Challenge: MtDetector: A High-performance Marine Traffic Detector at Stream Scale. In *DEBS '18: The 12th ACM International Conference on Distributed and Event-based Systems, June 25–29, 2018, Hamilton, New Zealand*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3210284.3220504

1 DEBS18 GC PROBLEM FORMULATION

In the 2018 DEBS Grand Challenge [3], the task is to predict the destination and arrival time given the spatio-temporal streaming data from vessels. The data is a sequence of tuples where each tuple contains the *ship ID*, *ship type*, *speed*, *longitude*, *latitude*, *course*, *heading*, *time stamp*, *departure port* and *draught*. A list of ports and a set of training data are provided for building machine learning models. The evaluation takes both the prediction accuracy (75%) and the system performance (25%) into account. The formula to calculate the accuracy of arrival port prediction for a trip is:

$$Accuracy = \frac{\text{The length of the } \textit{last} \text{ correctly predicted sequence}}{\text{The total number of tuples in a trip}}$$

The formula to calculate the accuracy of arrival time:

$$Accuracy = \frac{\sum |\text{Predicted arrival time - real arrival time}|}{\text{The total number of tuples}}$$

Here is an example demonstrating the accuracy calculation of port prediction: Assume a trip's destination is port A and there are 10 tuples in this trip. If the predicted sequence is $\{B, A, A, A, A, C, B, A, A, A\}$, the accuracy of the prediction is $\frac{3}{10} = 0.3$, even though the total number of correct labels is seven. Therefore, the evaluation metric is the key that makes the contest challenging. The accuracy value of arrival port prediction is only computed from the *earliest* correct point. One may generate 99% correct label prediction while making a wrong label near the end of the sequence can cause the final accuracy to drop to 0%.

2 ARRIVAL PORT PREDICTION

2.1 Arrival Port Neural Network Classifier

To predict the destinations of vessels, our idea is to build a neural network classifier to predict the arrival port. Because the port list is known, MtDetector builds a neural network classifier per port to predict the destinations of ships departing from the port. The idea comes from the observation that ships departing from the same port only arrive at a specific subset of ports. Thus, separating the models for ports can effectively reduce the solution space and improve the prediction accuracy. Next we select features that are useful for port prediction, including *ship type*, *ship position* (longitude and latitude), *speed*, *course*, and *offset of longitude and latitude from the ship's positions to all the ports*. The *ship type* is useful in the sense that ships with same type might follow the same route. The ship's position and offsets convey meaningful spatial information such as how long the ship has travelled and the distance between the ship and other ports. *Speed* is selected because a ship will gradually slow down when approaching its destination and *course* reflects the ship's intended route direction.

2.2 Incremental Majority Filter

A stable prediction result is critical for the accuracy of arrival port prediction. To prevent the prediction from changing frequently due to the noise such as ship drifting or wandering, we design an

*Both authors contributed equally to the paper

DEBS '18, June 25–29, 2018, Hamilton, New Zealand

ACM ISBN 978-1-4503-5782-1/18/06…$15.00
https://doi.org/10.1145/3210284.3220504

incremental majority filter algorithm to reduce the variation in predictions. For each trip, we record the predictions made by the neural network up to the current time stamp. Then we select the most frequently predicted port in the record as the new prediction. Consider the previous example whose predicted sequence from neural network is:

$$\text{Predicted sequence} = \{B, A, A, A, A, C, B, A, A, A\}$$

Then the sequence after applying the incremental majority filter becomes:

$$\text{Predicted sequence} = \{B, A, A, A, A, A, A, A, A, A\}$$

And the accuracy is increased from 30% to 90% in this case. The rationale behind this is we only accept the prediction change when there exists sufficient observations supporting the change. In above example, the occurrence of B and C are both less than A up to the current time stamp. Hence, they are treated as noise and being rejected by the filter. Although the algorithm does not guarantee to derive better accuracy after filtering, we find this algorithm does improve the accuracy notably in our experiments.

3 ARRIVAL TIME PREDICTION

Several research [4] [5] have shown the effectiveness of machine learning on maritime traffic arrival time prediction. MtDetector applies a Deep Neural Network (DNN)-based approach to estimate the relationships among variables and find a network that produces the best prediction accuracy.

3.1 Feature Selection

Feature selection plays the most important role in the solution quality of a DNN. Prior works assume data independence and hope the DNN to dig out important information from the data. However, we have found this insufficient for generating a descent result, primarily due to the large dependencies among data. For example, the position (latitude, longitude), speed, and heading of a moving ship all connect to each other between successive reports. Therefore, we consider the following ten features in our DNN: *time stamp*, *ship type*, *speed*, *longitude*, *latitude*, *course*, *cumulative distance*, *cumulative time*, *heading*, and *bearing*. Figure 1 shows the DNN structure of our default arrival time predictor.

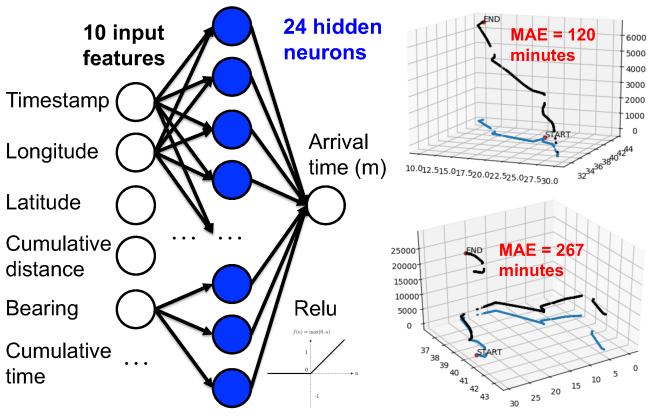

Figure 1: MtDtector's default arrival time predictor.

In addition to the features provided by the contest, we add three more features to our model, *bearing*, *cumulative distance*, and *cumulative time*. Bearing is defined as the angle between a ship's current position and the magnetic North. Cumulative distance is the total moving distance from a ship's departure to its current time stamp, measured over the earth's surface. Cumulative time is the total traveling time of a ship from departure to its current time stamp, measured in minutes. In fact, cumulative distance and cumulative time contribute a lot to the final accuracy value. Intuitively speaking, the larger the two values, the sooner a ship will arrive in its destination. These features are calculated on a *per trip* basis. A trip is defined as a single travel between two ports.

3.2 Model Selection

As ships belong to multiple types, it is difficult to have a universal model estimating ships' arrival times under distinct conditions, for example, vessel size, speed, draught, and shapes. To mitigate this problem, we apply different DNNs for different ship types. For each ship type, we conduct grid search in terms of number of layers, number of neurons, mini-batch size, and learning rate to obtain the best DNN structure. In fact, we also tried other machine learning techniques such as Recurrent Neural Network (RNN) and Logistic Regression. DNN turns out to outperform others. The data is split to two sets, 95% for training and 5% for testing/validation. Since the training set contains only a limited number of ship types, we generate a default DNN across all ship types. If during the online benchmarking one ship type is not found, the default DNN is used to predict its arrival time.

4 MTDECTOR ON THE HOBBIT PLATFORM

The MtDetector contains three parts: an adaptor, a command component and a task component. Figure 2 shows the system architecture of MtDtector. The Hobbit Platform uses RabbitMQ to create message queues for communication between systems and the adaptor is used to set up connections to the queues when the MtDetector activates. Once the queues are successfully connected, MtDetector launches a command component and a task component to handle the incoming messages from the command queue and task queue respectively.

MtDetector is highly parallel as every component is executed by an individual thread. The command component listens to the command queue and reacts on different system control commands such as notifying the task component when receiving a task generator finish signal. The task component listens to the task queue and makes predictions for incoming tasks. A task can be either predicting the arrival port or the arrival time of a ship and MtDetector identifies the type of task through examining the environment variables. Task components utilizes a thread pool to simultaneously handle multiple tasks to increase the throughput. When a message arrives, the task component extracts the task from the message and inserts the task into a work queue. The thread pool has several threads monitoring the work queue and a thread will be dispatched to process an awaiting task in the first-come-first-serve manner. The work thread forwards the prediction result to

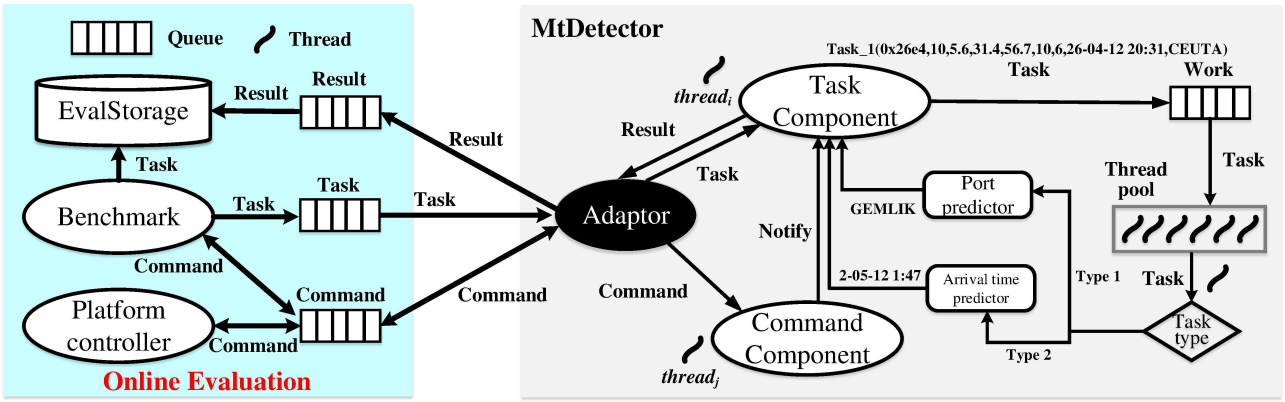

Figure 2: The system architecture of MtDetector. MtDetector consists of three components: an adaptor, a command component and a task component. Hobbit platform relies on queues to exchange data between systems, and the adaptor builds connections to those queues (task, command, result). Once the connections are set up, the command component and task component will be launched to handle the incoming messages. Both components are executed by individual threads to increase efficiency. When receiving a task message (i.e., ship data tuple), the task component extracts the task from the message and inserts the task into a work queue. A thread pool then dispatches a thread to process an awaiting task in work queue and send the result to the evaluation storage.

the task component after processing the task, and the task component sends both the task ID and the prediction to the evaluation module.

5 EXPERIMENTAL RESULTS

We discuss in this section the experimental results of MtDetector on two data sets, `debs18_training_fixed_3.csv` and `debs18_training_labeled.csv`, released by the official contest. The total number of ports is 40 in the Mediterranean sea.

5.1 Arrival Port Predictor

We first evaluate the port predictor on the given training data. The experiment is conducted on a single machine with 4 CPUs and 24 GB memory. We split the training data into trips based on the time stamp and departure/arrival ports and then categorize the trips based on their departure ports. For each port, we use the corresponding trips to build a neural network classifier with a hidden layer containing 90 neurons. The parameters of each neural network: batch size is 32, learning rate is 0.0003 and the number of epoch is 300. We take 95% of the tuples as training data set and 5% trips as the testing data set and report *the average of total correct predictions made by neural network*, *the average prediction accuracy without incremental majority filter* and *the average prediction accuracy with incremental majority filter*.

Table 1 shows the results of our port predictor. For most of the ports, our neural network classifier obtains high accuracy considering the number of total correct prediction. However, the accuracy drops significantly when being evaluated by the last correctly predicted sequence, for example, for port GEMLIK the accuracy decreases to 0.001 even 90% of the tuples are correctly predicted. This is expected as a wrong prediction zeros out the accuracy regardless of the past predictions. This issue is substantially mitigated after applying the incremental majority filter. It is shown that in most cases the incremental majority filter can keep the accuracy more closer to the number of total correct predictions, preventing the

Table 1: Results of Arrival Port Prediction

Port	Ratio of correct labels	Accuracy (w.o. IMF)	Accuracy (w. IMF)
ALEXANDRIA	0.5	0.5	0.5
AUGUSTA	0.8165	0.5768	0.7462
BARCELONA	0.9665	0.5764	0.9574
CARTAGENA	0.9931	0.9931	0.9882
CEUTA	0.7907	0.6713	0.6667
DAMIETTA	0.9327	0.0	0.8969
DILISKELESI	0.2442	0.0	0.0
FOS SUR MER	0.6991	0.006579	0.4458
GEMLIK	0.9296	0.001675	1.0
GENOVA	0.9579	0.6634	0.9449
GIBRALTAR	0.8598	0.3675	0.7877
HAIFA	0.9099	0.8049	0.866
ISKENDERUN	0.8304	0.8164	0.6954
LIVORNO	0.8636	0.4577	0.7744
MARSAXLOKK	0.975	0.7543	0.9574
MONACO	0.9724	0.7793	1.0
NEMRUT	0.959	0.808	0.9702
PALMA DE MALLORCA	0.8921	0.471	0.8832
PIRAEUS	0.9636	0.9178	0.9751
PORT SAID	0.9967	0.02455	1.0
TARRAGONA	0.8851	0.4215	0.844
TUZLA	1.0	1.0	1.0
VALENCIA	0.9451	0.9169	0.8934
VALLETTA	0.9921	0.9107	0.9862
YALOVA	1.0	1.0	1.0

accuracy drop caused by the wrong prediction. We believe this is due to the effective reduction of varying prediction.

5.2 Arrival Time Predictor

The second experiment is to evaluate the arrival time prediction of MtDetector. Table 2 lists our DNN model parameters for arrival time prediction. In each column, "Type" denotes the ship type, "Layer" denotes the network structure, "Lrate" denotes the learning rate, "Decay" denotes the rate we reduce the learning rate for every 10000 epochs, "B" denotes the mini-batch size used during training, "Epoch" denotes the number of training epochs, and "MAE" represents the Mean Absolute Error (MAE) in minutes across the whole data set. We evaluated our model on an emulated environment of two nodes using the DtCraft system [1], where one node sends the ship data and another node performs the prediction. Each node has 4 CPUs and 28 GB RAM

Table 2: Results of Arrival Time Prediction

Type	Layer	Lrate	Decay	B	Epoch	MAE (m)
0	10x20x1	0.01	0.95	32	7000	267.886
20	10x10x1	0.01	0.95	16	5000	124.287
30	10x8x1	0.01	0.95	32	7000	263.289
32	10x14x1	0.01	0.95	32	8000	98.9761
34	10x20x1	0.01	0.95	64	9000	90.0325
36	10x30x1	0.01	0.95	32	9000	300.879
37	10x18x1	0.01	0.95	16	8000	621.656
51	10x32x1	0.01	0.95	32	10000	82.5672
52	10x30x1	0.01	0.95	16	7000	502.426
60	10x32x1	0.01	0.95	64	7000	126.394
66	10x12x1	0.01	0.95	64	4000	51.90
69	10x30x1	0.01	0.95	32	9000	186.153
70	10x24x1	0.01	0.95	32	50000	827.382
71	10x22x1	0.01	0.95	64	50000	392.387
72	10x30x1	0.01	0.95	64	50000	83.3375
73	10x12x1	0.01	0.95	16	50000	38.6481
74	10x26x1	0.01	0.95	32	50000	113.232
76	10x30x1	0.01	0.95	64	8000	19.4973
79	10x30x1	0.01	0.95	64	8000	278.228
80	10x22x1	0.01	0.95	32	50000	443.919
81	10x16x1	0.01	0.95	16	50000	543.749
82	10x32x1	0.01	0.95	64	50000	13.1808
83	10x20x1	0.01	0.95	32	50000	35.3363
84	10x20x1	0.01	0.95	16	10000	26.4493
85	10x20x1	0.01	0.95	32	7000	24.7699
89	10x22x1	0.01	0.95	64	9000	189.787
90	10x24x1	0.01	0.95	16	10000	137.897
99	10x8x1	0.01	0.95	16	10000	51.2497

The results indicate two strengths of MtDetector: (1) Having different models for ship types can effectively estimate the arrival time with MAE less than one day. In many cases, the MAE can be less than 1 hour. (2) Our feature selection method efficiently reduces the DNN size. One layer is sufficient for all cases, which would otherwise take more than three layers to generate similar results by using only the raw features.

In addition to DNN, we have tried Recurrent Neural Network (RNN)-based regression to estimate the arrival time. RNN is a popular method that has shown great promise in many Natural Language Processing (NLP) tasks. The idea is to extract *trips* from each ship and use a trip as the basic unit during the training. A trip is a route ordered by time stamp between two ports.

Table 3: Comparison between DNN and RNN

Method	Layer	Lrate	Train	MAE (m)
DNN	10x32x1	0.01	>10 hr	234.217
RNN	10x32x1	0.01	1 hr	767.044

Unfortunately, RNN cannot generate a good quality result as DNN. As presented in Table 3, the solution quality of RNN in terms of MAE is much worse than DNN in an example data set. Also, the complexity to train a RNN is much higher than a DNN (>10 hr vs 1 hr). With the information provided in the contest dataset, it is very difficult to correctly identify trips out of each ship. Even though there are heuristics to mitigate this problem, most of them compromise on accuracy. Besides, RNN faces the problem of *vanishing gradient* and *exploding gradient* problem in training a long trip. These issues make it critical to apply RNN to solve this problem.

6 CONCLUSION

In this paper, we introduce MtDetector, a high-performance marine traffic detector to predict the arrival port and arrival time of vessels. For arrival port prediction, we build a neural network classifier for each port which effectively reduces the solution space. Furthermore, considering the evaluation method, we develop an incremental majority filter to enhance the prediction accuracy. For arrival time prediction, we propose to build deep neural network regressors based on the ship type as ships with the same type have more similar characteristics. The experimental results demonstrate the high prediction accuracy of MtDetector in both the port and time prediction.

7 ACKNOWLEDGMENT

We appreciate all reviewers' efforts on reviewing this work. Special thanks go to contest organizers (Zbigniew Jerzak, Pavel Smirnov, Martin Strohbach, Holger Ziekow, and Dimitris Zissis) for their hard work on helping contestants resolve various technical issues throughout the contest.

REFERENCES

[1] DtCraft. http://dtcraft.web.engr.illinois.edu/.
[2] T.-W. Huang, C.-X. Lin, and Martin D. F. Wong. DtCraft: A High-performance Distributed Execution Engine at Scale. In *IEEE TCAD*, 2018.
[3] Vincenzo Gulisano, Zbigniew Jerzak, Pavel Smirnov, Martin Strohbach, and Holger Ziekow. The DEBS 2018 grand challenge. In *Proceedings of the 12th ACM International Conference on Distributed and Event-based Systems, DEBS 2018, Hamilton, New Zealand, June 25-29, 2018*, 2018.
[4] Andrius Daranda. A neural network approach to predict marine traffic. Technical Report MII-DS-07T-16-9-16, Vilnius University, Institute of mathematics and informatics, Lithuania, Oct 2016.
[5] Ioannis Parolas. ETA prediction for containerships at the Port of Rotterdam using Machine Learning Techniques. Master's thesis, Delft University of Technology, the Netherlands, 2016.