International Journal of Internet, Broadcasting and Communication Vol.13 No.4 135-142 (2021)
http://dx.doi.org/10.7236/IJIBC.2021.13.4.135

IJIBC 21-4-17

A Human Movement Stream Processing System for Estimating Worker Locations in Shipyards

Dat Van Anh Duong and Seokhoon Yoon*

Ph.D Student, Department of Electrical, Electronic and Computer Engineering, University of Ulsan, Korea
Associate Professor, Department of Electrical, Electronic and Computer Engineering, University of Ulsan, Korea
mrdvad11@gmail.com, seokhoonyoon@ulsan.ac.kr

Abstract

Estimating the locations of workers in a shipyard is beneficial for a variety of applications such as selecting potential forwarders for transferring data in IoT services and quickly rescuing workers in the event of industrial disasters or accidents. In this work, we propose a human movement stream processing system for estimating worker locations in shipyards based on Apache Spark and TensorFlow serving. First, we use Apache Spark to process location data streams. Then, we design a worker location prediction model to estimate the locations of workers. TensorFlow serving manages and executes the worker location prediction model. When there are requirements from clients, Apache Spark extracts input data from the processed data for the prediction model and then sends it to TensorFlow serving for estimating workers' locations. The worker movement data is needed to evaluate the proposed system but there are no available worker movement traces in shipyards. Therefore, we also develop a mobility model for generating the workers' movements in shipyards. Based on synthetic data, the proposed system is evaluated. It obtains a high performance and could be used for a variety of tasks in shipyards.

Keywords: *Mobility Model, Location Prediction, Location Data Stream, Data Frame, Stream Processing System.*

1. Introduction

In this work, we propose a human movement processing system for estimating the locations of workers in shipyards. Estimating worker locations can be valuable information for internet of thing (IoT) services, wireless networks, and automated systems [1, 2]. For example, knowing the estimated next locations of workers allows us to select the optimal worker for transmitting messages over a wireless network, and quickly find and rescue workers when accidents or industrial disasters occur. The human movement stream processing system is built based on Apache Spark [3] and TensorFlow serving [4].

Specifically, Apache Spark is a computing platform [3]. It is optimized for the execution of big data applications and provides an appropriate framework for continuously querying data without requiring access to system storage/disk. Therefore, Spark is a suitable framework for iterative processing, batch processing, streaming, and interactive queries. In this work, workers' location data is streamed to Apache Spark by using

Manuscript Received: October. 2, 2021 / Revised: October. 5, 2021 / Accepted: October. 10, 2021
Corresponding Author: seokhoonyoon@ulsan.ac.kr
Tel: +82-52-259-1403, Fax: +82-52-259-1687
Associate Professor, Department of Electrical, Electronic and Computer Engineering, University of Ulsan, Korea

the message queuing telemetry transport protocol (MQTT) [5]. Spark receives location data and partitions it into batches. Each batch is processed to update to a data frame. In this work, we also design a worker location prediction model to estimate worker locations via TensorFlow Serving [4]. TensorFlow Serving is a flexible, high-performance serving solution for machine learning models. That is optimized for industrial operation. TensorFlow Serving enables faster deployment of new algorithms and experiments while maintaining the server architecture and APIs that are already in place. When Apache Spark receives a request from a client, it extracts the input data for the prediction model from the data frame. Then, the input data and the request are transmitted to TensorFlow serving for estimating worker locations. Finally, Apache Spark receives the result from TensorFlow serving and returns it to the client.

The movement traces of workers are required for training the prediction model and validating the proposed system. However, there is no available movement dataset of workers in shipyards. Therefore, we proposed a mobility model for workers in shipyards. Numerous human movement models have been studied [6, 7], but the majority of them focus on people's daily movements. This cannot accurately reflect the movements of shipyard workers. For example, the urban context-aware mobility model [6] has been studied to investigate more realistic flight and pause time distributions. The mobility model described in [7] considers both social relationships between people and the characteristics of human movement. However, such mobility models take into account only the general situation of daily human movement in urban areas. None of them are specifically concerned with the characteristics of shipyard worker movement. In our proposed model, we consider the workflow when generating workers' movements. First, workers are classified into types. Workers belonging to the same type exhibit similar movement characteristics (e.g., movement speed and pause time). Then, each type is separated into teams. A team's members share the same workflow and workspace. Workers move in accordance with their workflow.

Finally, to verify the proposed system, the synthetic data is used for training the worker location prediction model. An application is developed to stream workers' location data to Apache Spark. Then, the estimated results are obtained with the synthetic data.

The rest of this paper is organized as follows. First, we describe how to generate worker movements in shipyards in Section 2. Then, Section 3 presents the human movement stream processing system. The experimental results are discussed in Section 4. Finally, we conclude this paper in Section 5.

2. Generating Worker Movements in Shipyards

In this section, the movement area of workers is described. Then, a mobility model of workers in shipyards is proposed based on their workflow.

In this work, a shipyard layout in [8] is used as the movement area of workers. There are multiple subareas in a shipyard. In this study, we focus on the movement of workers inside a subarea (i.e., unit assembly (UA)). The area of UA is 310m×400m and it is divided into 32 blocks. Each block has the size of 77.5m×50m and is partitioned into 20 units.

Workers in UA are categorized into a variety of types. We consider that workers of a particular type perform the same type of tasks. For instance, workers in type 1 welds and cuts steel. Workers in type 2 paint ships. Workers in type 3 are those who operate forklift trucks. Workers within a type will move at a similar speed and pause time.

Now, we discuss the mobility model of workers. Specifically, workers in a type are grouped into teams. At the beginning of a day, each team is randomly assigned to one of 32 blocks in UA. The team members will work in the designated block throughout the day. First, workers in the team randomly select a unit in the assigned block to start their work. An example of the workflow in one of 32 blocks in the area of UA is shown in Figure 1. In the figure, worker w_j starts his/her work at unit $u1$ and then follows the workflow to move through units $u2, u3, \ldots, u20$ and complete tasks at each unit. After completion of the task at unit $u20$, worker w_j moves to unit $u1$ and starts a new process following the workflow.

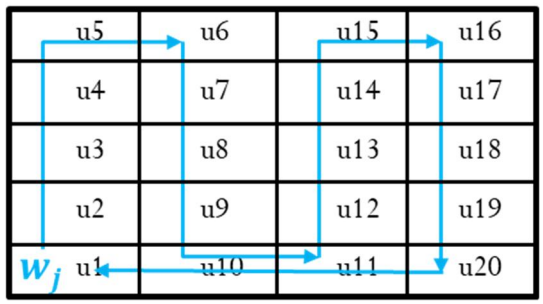

Figure 1. An example of workflow in a block.

In reality, we observe that once the pause time has expired (i.e., the worker has completed the job) at the current unit, the worker has a high probability of visiting the next unit in the workflow and a low probability of visiting other units. For instance, worker j has completed his or her job at unit i. To reflect this realistic movement, we define p_{next}^j as the probability that workers will visit the next unit in the workflow, and it is set to a high value. Let $\boldsymbol{S}^i$ denote the set of neighbor units of unit i (e.g., $\boldsymbol{S}^{u1} = \{u9, u10\}$). The number of the same team workers of worker j in unit i is defined as n_l^j. The probability that worker j visits unit l in $\boldsymbol{S}^i$ is defined as p_l^j. It is calculated as follows:

$$p_l^j = \left(1 - p_{next}^j\right) \times \frac{e^{n_l^j}}{\sum_{k \in \boldsymbol{S}^i} e^{n_k^j}} \qquad (1)$$

Equation (1) indicates that worker j prefers to visit a neighbor unit with a high value of n_l^j. This reflects the real context that workers tend to go to the place where a lot of their team members are.

Finally, the worker randomly selects a position to visit within the selected unit. The movement speed and pause time of workers are based on real mobility traces [9-10].

3. The Human Movement Stream Processing System

In this section, we first present how workers send locations to Apache Spark via MQTT protocol [5]. Then, the processing of location data in Apache Spark is described. Finally, estimating worker location with TensorFlow serving is discussed. The worker movement stream processing system is shown in Figure 2.

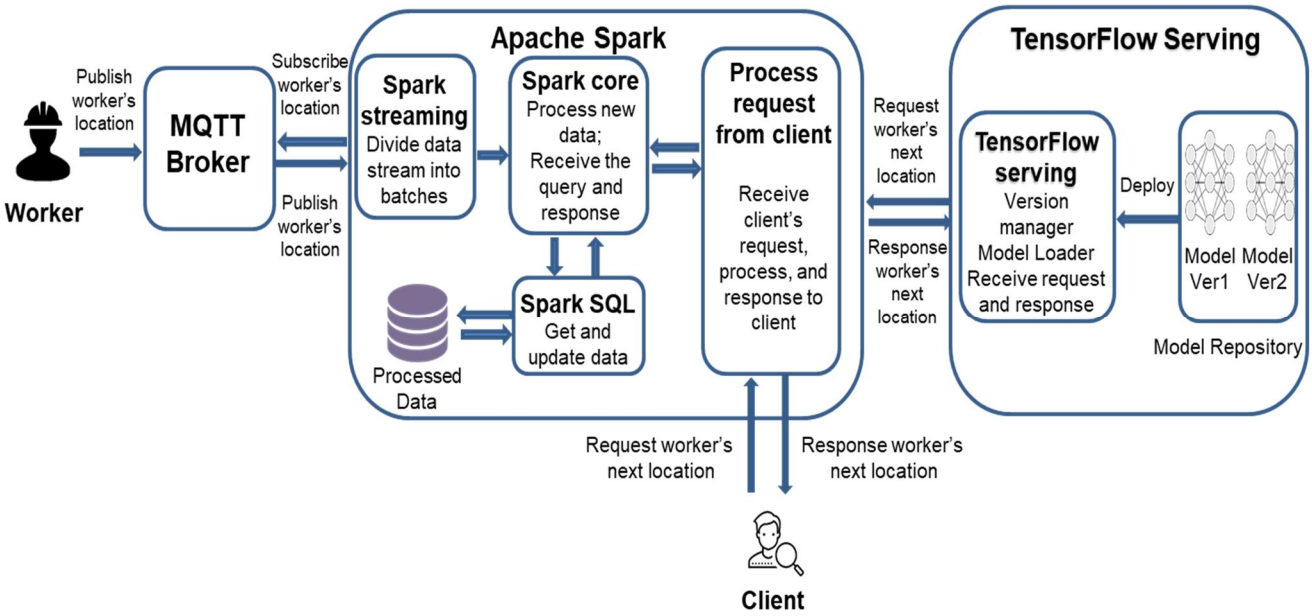

Figure 2. The human movement stream processing system.

3.1 Sending Worker's Location with MQTT Protocol

MQTT protocol is used for workers to send their location data to Apache Spark. MQTT is a lightweight, publish-subscribe network protocol for exchanging messages between devices. This protocol is extensively used in industrial networks because of its scalability and support for dynamic application topologies. These characteristics are achieved by decoupling publishers from subscribers, which enables the addition of new data sources or the replacement of existing modules. Client connections are always handled by a broker in MQTT. A client of MQTT could be either a publisher or a subscriber. A publisher is a client who sends messages to the broker on a particular topic. A subscriber will subscribe to topics on MQTT broker. When the broker receives messages from publishers about a topic, it forwards them to subscribers who have subscribed to the topic. As shown in Figure 2, in our system, workers are publishers who publish information to the MQTT broker about the topic of the worker location. Apache Spark is a subscriber to the worker location topic. When a worker sends the location to the MQTT broker, the broker will transmit the location data to Apache Spark.

3.2 Processing Data Stream by Using Apache Spark

After receiving location data from the MQTT broker, this subsection discusses how Apache Spark processes received data and client requests.

Apache Spark is a framework for performing distributed computation on big data by using in-memory primitives. This platform allows user programs to put data into memory and frequently query it, which makes it a great tool for real-time and iterative processing. Spark is built on distributed data structures known as resilient distributed datasets (RDDs). Operations on RDDs automatically partition tasks and maintain the location of persistent data. Additionally, RDDs are a flexible tool that enables programmers to store intermediate results in memory as a data frame or on disk for later reuse, as well as adjust partitioning for optimal data placement.

By processing data in batches, Spark streaming enables us to use Spark's API in streaming environments.

Each batch contains multiple RDDs that were received from the MQTT broker during a time window. As shown in Figure 2, Spark streaming receives the worker's location data then divides data into batches. The steps for data processing are applied to each batch in Spark core. In our system, an RDD contains information of a record from the workers. For instance, this includes the identification of the worker, the worker type, the worker team, the recording time, the unit (where the worker is), and the location coordinates. From the recording time, information about the day in the week and the time slot of the day is obtained. Other necessary information is also gathered such as the worker identification and the unit.

The collected information is then processed by Spark SQL. The data frame, a new data structure for structured data, is introduced in Spark SQL and supports the use of SQL queries in Spark programs. It allows us to update and get data from a data frame. In this work, the collected data (i.e., worker identification, the time slot of the day, the day in the week, and the unit) is maintained in a data frame. When a new batch of data is coming, the data frame is also updated.

3.3 Estimating Worker's Location with TensorFlow Serving

In this subsection, we first design a worker location prediction model. Then, estimating the worker's location by using TensorFlow Serving is discussed.

3.3.1 The worker location prediction model

To predict the next location of workers, a worker location prediction model is proposed. That is a recurrent neural network (RNN)-based model using long short-term memory (LSTM) cells [11, 12]. The model will take worker locations at current and previous time slots as an input and then outputs the location of workers in the very next time slot.

The architecture of the proposed model is shown in Figure 3. Specifically, a one-hot vector is used to distinguish workers. Let a_w^I denote the one-hot vector that indicates worker identification of worker w. In this model, we consider the working time from 8:00 to 18:00. The working time is divided into time slots of 15 minutes. The one-hot vector which represents the time slot *t* in a day is denoted as a_t^T. The day in the week is also considered and let a_t^D be the one-hot vector that indicates the day in the week of time slot t. The one-hot vector that indicates the location (i.e., the unit) of worker w at time slot t is defined as $a_{w,t}^L$. The input is the information of workers from time slot $(t-k+1)$ to time slot t (e.g., $x_{w,t} = \{a_w^I, a_t^D, a_t^T, a_{w,t}^L\}$). Let $y_{w,t+1}$ define the output, which is a vector with elements that are probabilities that worker w visits locations at time slot $t+1$. For example, with $y_{w,t+1}$ = [0.1, 0.05, 0.01, …, 0.3], the predicted probability for visiting unit 1 is 0.1; the predicted probability for visiting unit 2 is 0.05.

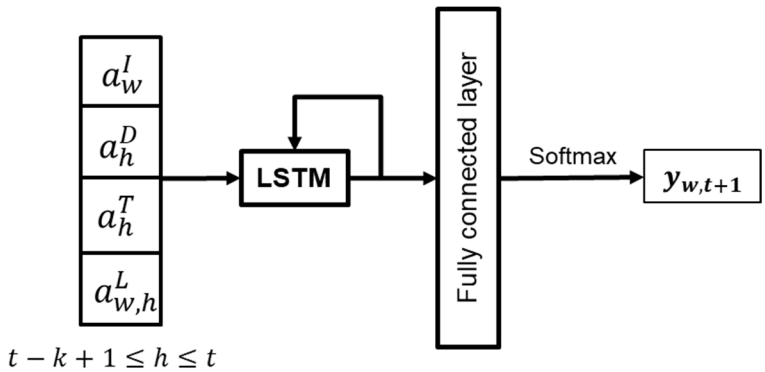

Figure. 3 The worker location prediction model.

3.3.2 Running the worker location prediction model with TensorFlow serving

TensorFlow Serving is a machine learning serving platform that enables on-demand model inference. It is widely used in industrial environments and provides high-performance serving. TensorFlow serving decouples prediction model from the rest of the system. That allows switching model versions and simultaneously serving multiple models. TensorFlow Serving provides low latency serving and it is self-service, requiring users to deploy and operate it on local (or cloud) infrastructure in order to perform predicting.

In our system, TensorFlow serving is deployed on a server. It manages worker location prediction models and versions of the models. Specifically, the proposed worker location prediction model is put in TensorFlow serving and Docker is used as the execution environment of the prediction model. When Apache Spark sends a request and data (the input data for the prediction model) to TensorFlow serving, TensorFlow serving will check that which model and which version are required. Then, the required model with a suitable version is used for estimating the worker's next location by using the input data from Apache Spark.

3.4 Processing client's requests

Now, we discuss how to handle the client's request. A thread is started in Apache Spark to receive and process client requests. Clients may submit requests via any IoT protocol (e.g., MQTT and TCP socket). The TCP socket protocol is used to communicate between Spark and clients in this work. When Apache Spark receives a request from a client for the location of a worker, it extracts the necessary information from the data frame (i.e., the worker identification, the unit, the day in the week, and the unit) by using Spark SQL. The data is then sent to TensorFlow serving, which is used to estimate the worker's next location. Finally, the estimating result is sent to Apache Spark and Apache Spark responds to the client.

4. Experimental Results

In this work, the mobility model of workers generates data for training the worker location prediction model. A program is used to send workers' location data to the MQTT broker. The sent location data is the synthetic data from the mobility model. The proposed system will process and respond to the client base on estimating locations of workers with those data.

4.1 Experimental Setup

In the mobility model of workers, p_{next}^{j} is set to 0.98. The number of workers is set to 200. The workers are partitioned into 4 types (50 workers in a type). Each type has 5 teams (10 workers in a team). The movements of workers are generated for 90 days and 10 hours per day. Based on the real mobility trace [9-10], the pause time of workers is set to follow a truncated power-law distribution with a range of values from 6 to 360 minutes and the movement speed follows a log normal distribution, $Lognormal(-0.684, 0.97^2)$.

From 90 days of the synthetic data, we use 54 days for training, 18 days for validating, and 18 days for testing the worker location prediction model. The input and output data format of the prediction model is presented in Table 1. The input ($x_{w,t}$) includes the identification of worker (a_w^I), the day in the week (a_t^D), the time slot (a_t^T), and the location of worker ($a_{w,t}^L$). Those input features are represented by one-hot vectors. The output is a vector ($y_{w,t+1}$) with elements that are predicted probabilities that worker w visits locations at time slot $t+1$. The detailed settings of the prediction model are shown in Table 2. The number of LSTM cells is set to 16. Then, the number of lookback time steps is 4. Finally, the number of fully connected neurons is set to 1024.

Table 1. Input and output data format of the worker location prediction model.

$x_{w,t} = \{ a_w^I, a_t^D, a_t^T, a_{w,t}^L \}$				$y_{w,t+1}$
a_w^I	a_t^D	a_t^T	$a_{w,t}^L$	Vector (1x640)
One-hot vector (1x200)	One-hot vector (1x7)	One-hot vector (1x41)	One-hot vector (1x640)	

Table 2. Settings of the worker location prediction model.

Parameter	Value
Cell units	**16**
Lookback time steps	**4**
FC neurons	**1024**

The Apache Spark and TensorFlow serving are running on a machine. The program which simulates streaming location from 200 workers is running on a machine, and 10 clients send the request from another machine. Those machines have the same configuration as follows:

- Intel(R) Xeon(R) CPU E3-1240 V2 @ 3.4Ghz (8 CPUs)
- 8GB 1600 MHz DIMM DRAM
- 1TB Hard Disk Drive
- 90 Mbps network connection

4.2 Experimental Results

Clients could get the worker's next location for top-1 accuracy, top-3 accuracy, and top-5 accuracy from the proposed system. The accuracy of the proposed system with the synthetic data is shown in Table 3. As can be seen in Table 3, the system achieves 68.52% accuracy point with top-1, 97.26% accuracy point with top-3, and 98.67% accuracy point with top-5. In general, the accuracy increases from top-1 to top-5. The obtained results also indicate that the position of workers is accurately estimated on top-3. That information is valuable for various applications in shipyards. The proposed system is run well with the configuration of 10 clients and 200 workers.

Table 3. Average accuracy of the proposed system on the synthetic data

Top – 1 accuracy	Top – 3 accuracy	Top – 5 accuracy
68.52	97.26	98.67

5. Conclusion

In this work, we propose a human movement stream processing system for estimating worker locations in shipyards. First, a mobility model is designed to generate the movements of workers in shipyards. Then, a worker location prediction model is developed to estimate the location of workers in shipyards. The prediction

model is trained on the synthetic data from the mobility model. Apache Spark is used to process location data streams from workers, receive the client requests, get the estimation results from TensorFlow serving, and response to clients. TensorFlow serving manages prediction models and runs prediction models when receives the request from Apache Spark. The proposed system achieves high accuracy and performs well with a large number of workers and clients. In future work, to analyze the performance of the system in the real situation, we plan to collect the real data of worker movements in shipyards and run the system with the real data.

Acknowledgement

This research was supported by Institute of Information & communication Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (2020-0-00869, Development of 5G-based Shipbuilding & Marine Smart Communication Platform and Convergence Service). This work was also supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (2019R1F1A1058147).

References

[1] S. H. Yoon, K. S. Lee, J. S. Cha, V. Mariappan, M. W. Lee, D. G. Woo, and J. U. Kim, "Indoor Surveillance Camera based Human Centric Lighting Control for Smart Building Lighting Management," *International Journal of Advanced Culture Technology (IJACT)*. Vol. 8, No. 1, pp. 207-212, 2020.
DOI: doi.org/10.17703/IJACT.2020.8.1.207

[2] K. R. Lee, "Development of Auto-tuning Temperature controller with Multi-channel," *The Journal of the Convergence on Culture Technology*. Vol. 4, No. 4, pp. 419-427, 2018.
DOI: doi.org/10.17703/JCCT.2018.4.4.419

[3] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, J. F. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," *In 9th Symposium on Networked Systems Design and Implementation*, pp. 15-28. 2012.

[4] C. Olston, N. Fiedel, K. Gorovoy, J. Harmsen, L. Lao, F. Li, V. Rajashekhar, S. Ramesh, and J. Soyke, "Tensorflow-serving: Flexible, high-performance ml serving," *arXiv preprint arXiv:1712.06139*, Dec 2017.

[5] S. A. Shinde, P. A. Nimkar, S. P. Singh, V. D. Salpe, and Y. R. Jadhav, "MQTT-message queuing telemetry transport protocol," *International Journal of Research*, Vol. 3, No. 3, pp.240-244, 2016.

[6] X. Kang, L. Liu, H. Ma, and D. Zhao, "Urban context aware human mobility model based on temporal correlation," *In 2017 IEEE International Conference on Communications (ICC)*, pp.1-6, May 2017.
DOI: doi.org/10.1109/ICC.2017.7997157

[7] D. V. A. Duong, and S. Yoon, "SRMM: A social relationship-aware human mobility model," *Electronics*, Vol. 9, No. 2, p. 221, Feb 2020.
DOI: doi.org/10.3390/electronics9020221

[8] Y. J. Song and J. H. Woo, "New shipyard layout design for the preliminary phase & case study for the green field project," *International Journal of Naval Architecture and Ocean Engineering*, Vol. 5, No. 1, pp.132-146, Mar 2013.
DOI: doi.org/10.3744/jnaoe.2013.5.1.132

[9] D. Kotz, T. Henderson, T. Abyzov, and J. Yeo, CRAWDAD dataset dartmouth/campus (v. 2009-09-09), Sep 2009. *crawdad.org/dartmouth/campus/20090909*.

[10] I. Rhee, M. Shin, S. Hong, K. Lee, S. Kim, and S. Chong, CRAWDAD dataset ncsu/mobilitymodels (v. 2009-07-23), Jul 2009. *crawdad.org/ncsu/mobilitymodels/20090723.*

[11] L. R. Medsker, and L. C. Jain. "Recurrent neural networks: Design and Applications," Boca Raton: CRC Press, 1999.

[12] S. Hochreiter, and J. Schmidhuber, "Long short-term memory," *Neural computation*, Vol. 9, No. 8, pp.1735-1780, Nov 1997.
DOI: doi.org/10.1162/neco.1997.9.8.1735