Mercury: Metro Density Prediction with Recurrent Neural Network on Streaming CDR Data

Victor C. Liang §1 Richard T. B. Ma §‡2 Wee Siong Ng †3 Li Wang §4 Marianne Winslett ‡5
Huayu Wu †6 Shanshan Ying§7 Zhenjie Zhang§8

§*Advanced Digital Sciences Center, Illinois at Singapore Pte. Ltd., Singapore*
1,4,7,8{chen.liang, wang.li, shanshan.y, zhenjie}@adsc.com.sg
† *School of Computing, National University of Singapore, Singapore*
2tbma@comp.nus.edu.sg
† *Institute for Infocomm Research, Singapore*
3,6{huwu, wsng}@i2r.a-star.edu.sg
‡ *Department of Computer Science, University of Illinois at Urbana Champaign, USA*
5winslett@illinois.edu

***Abstract*—Telecommunication companies possess mobility information of their phone users, containing accurate locations and velocities of commuters travelling in public transportation system. Although the value of telecommunication data is well believed under the smart city vision, there is no existing solution to transform the data into actionable items for better transportation, mainly due to the lack of appropriate data utilization scheme and the limited processing capability on massive data. This paper presents the first ever system implementation of real-time public transportation crowd prediction based on telecommunication data, relying on the analytical power of advanced neural network models and the computation power of parallel streaming analytic engines. By analyzing the feeds of caller detail record (CDR) from mobile users in interested regions, our system is able to predict the number of metro passengers entering stations, the number of waiting passengers on the platforms and other important metrics on the crowd density. New techniques, including geographical-spatial data processing, weight-sharing recurrent neural network, and parallel streaming analytical programming, are employed in the system. These new techniques enable accurate and efficient prediction outputs, to meet the real-world business requirements from public transportation system.**

I. Introduction

Public transportation is one of the most important components in a smart city. With the growing big data processing and analytical power, the public transportation system is expected to support *fast response* or even *pre-response* to emergent events, e.g. road accidents and traffic burst, in order to improve transportation experience with commuters. To fulfil the vision of new generation of public transportation system in smart city, it is necessary and crucial for decision makers to understand the status of the transportation system, and foresee the dynamics of the system in the future. While public transportation system generates massive data on the fly, e.g. entrance records of metro stations, it is sometimes essential to include external source of information to capture the changes in smart city in time, which may provide additional and important knowledge. To address such challenges, information technologies are employed to enhance the utility of the prediction module in the system, with a robust and powerful data processing system mixing both internal and external data in an efficient way.

Telecommunication data is believed to be the most relevant source of external information for public transportation system. Generally speaking, telecommunication companies receive updates from all active mobile phone users, including their locations, velocities, etc. In Fig. 1, we present a diagram to illustrate the connection between telecommunication data and public transportation system. The database in public transportation system maintains the records of commuters when they travel with metros, buses and taxis, but does not have sufficient information to understand the behaviors of the commuters before they are physically in the system. The information of *potential commuters*, however, is crucial for the system to build their prediction functionalities. A common example is the heavy traffic load when a sports event finishes, by which tens of thousands of attendants may flood to the public transportation system. Without additional data from other sources, it is nearly impossible for public transportation administrator to predict such unexpected passenger burst. Telecommunication data, on the other hand, provides a thorough coverage on both commuters and potential commuters, which can be regarded as uniform samples over the potential population of public transportation system.

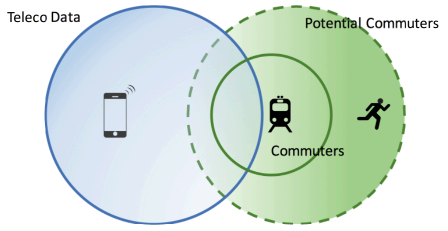


Fig. 1. The overlap of data coverage between telecommunication company and public transportation system.

In this paper, we present our new online prediction system, called *Mercury*, which estimates the crowd density of passengers in metro stations by real-time analysis over caller detail record (CDR) data from mobile phone users. Caller detail record (CDR) is used to track the behaviors of mobile phone users, when their phones communicate with the tower of base station, during phone call, messaging and Internet connection. Mercury provides online data processing and analytical service based on the online CDR data from a local communication company in Singapore, such that the system reports new crowd density prediction on, 1) the number of new passengers entering the metro stations; 2) the number of waiting passengers on the platforms; and 3) overcrowd alerts to metro administrators. To the best of our knowledge, this is the first attempt to directly transform telecommunication data into values for public transportation system in smart city.

There are a number of technical challenges in Mercury. Firstly, CDR data itself does not distinguish commuters from other mobile phone users. Even if a mobile phone user is physically close to a metro station, she/he could be a worker or resident in a nearby building, the presence of which may affect the accuracy of the prediction module. Secondly, the complexity of the dependency between the metro crowd density and nearby mobile phone population is extremely high, especially on highly varying latencies on conversions from potential commuters to actual commuters. In particular, different groups of users, such as diners in shopping malls and bus passengers in transfer, may come into the metro station at completely different rates. Thirdly, powerful neural network models, despite of their strong learning capability, are not directly applicable to spatial-temporal domain, because of the discrete neuron used on the input layer. Finally, it is also challenging to support data processing and analytics in real time, when there are thousands of learning tasks, proportional to the number of metro stations and bus stops, are registered and run in the system in parallel.

We tackle all these technical challenges by a number of novel approaches tailored for our system architecture and applications. To highlight the research outcomes used in the system, we list the main technical contributions below:

1) We design the first ever prediction system to evaluate the passenger density of metro stations based on streaming analytics over CDR data in real time.
2) We propose the first recurrent neural network model for sequential data from spatial-temporal domains and provide carefully designed initialization methods to improve the effectiveness of neural network training.
3) We devise sophisticated parallel streaming analytical algorithms over cloud-based streaming processing platform, to support huge online data analytics in Mercury.

II. Related Work

Time series prediction methods exist in three classes in terms of when learning stage starts, namely eager learning, lazy learning and semi-lazy learning. The eager learning approach usually has a pre-preprocessing step and trains a model for prediction. Stochastic Models [5], Artificial Neural Networks (ANNs) [2], [6] Support Vector Machines (SVM) [15], and Gaussian Process [16], [13] are four schools of models in eager learning. The lazy learning approach typically waits until a query series arrives, and then fetches k time series similar to the query (i.e, k-nearest neighbour(kNN)) and does some simple computation, such as mean, over the query result [3], [4]. Comparing these two approaches, the eager leaning approach suffers from high computational cost while the lazy approach stands low accuracy. Therefor, the semi-lazy approach is proposed as a trade off. It employs sophisticated models, such as Kalman Filter Gaussian Process Regression [17], on kNN query result. SMiLer [19] proposes the first semi-lazy framework for time series prediction, using Gaussian Process as the training model, accelerated by GPU. None of these techniques, however, could solve the density prediction problem, as the days that sensors monitoring the number of people in public area are yet to come.

Recurrent neural network is proposed in early 90s last century [18]. Although the model is powerful, the unmatched computer power and lack of optimization techniques at that time limit the usage of the model on real problems. In recent years, RNN has attracted attentions in both academia and industry, with the new theoretical advances on network structure [9], [10] and better optimization [12]. RNN is now widely used in a variety of areas in computer science, to support models on sequential data domain. In [11], Liu et al. build language models with RNN, to facilitate more accurate machine translations. In [14], Sutskever et al. employ RNN to generate meaningful text sequence, which is later used in Google for image capture generation.

A number of new parallel stream processing systems have emerged in last few years, such as Spark Stream, Storm and Heron. However, none of the systems listed above can fully satisfy the elasticity requirements to handle the varying workload of telecommunication data stream. We finally choose to use Resa [1], which is our in-house parallel stream processing engine with new technologies on scheduling [8] and migration [7].

III. System Design and Techniques

A. Problem Formulation

In Mercury, there are two different data sources, telecommunication data and metro data. We only use *caller detail records* from telecommunication data in prediction module. While different telecommunication company may adopt different schemas for their CDR table, we present an example in Tab. I, with a general schema covering all important information recorded by any telecommunication company. There are five attributes in each caller detail record. Specifically, the *user id* is an identifer of the mobile phone user associated with the record, *time* is the event time on the phone, *longitude* and *latitude* tell the location of the mobile phone and *action* indicates the type of events with the record. There are four categorical event types, including phone calling (CAL), text messaging (SMS), multimedia messaging (MMS) and Internet access (INT).

TABLE I
EXAMPLE TABLE OF CDR IN TELECOMMUNICATION COMPANY

User ID	Time	Longitude	Latitude	Action
1001	8:00:00am	103.737459	1.326909	SMS
1002	8:00:01am	103.737512	1.327108	CAL
1003	8:00:03am	103.741002	1.339921	SMS
1004	8:00:03am	103.738199	1.331231	INT

Based on the schema, it is clear that the telecommunication system generates a record only when an event of specified type happens on the mobile phone. The CDR data consists of sparse samples on the trajectories of mobile phone users. In our dataset, for example, the average number of records from an individual mobile phone user in 24 hours is 141. This property increases the difficulty of prediction module over CDR data. An aggregation is necessary on population level, in order to eliminate the randomness of event recording in the CDR data. On the other hand, the metro data consists of entrance and exit records of metro passengers. The metro data is only used in offline prediction training. The online analytics in Mercury do receive any real-time update from metro stations.

Given the two data sources, the target problem for Mercury is to provide real-time prediction service on two measurements on the crowd density of metro stations. The first measurement is the *number of new metro passengers* to particular metro stations, while the second is the *number of passengers waiting on platforms* of the metro stations.

B. System Architecture

There are three major modules in the system, including *Preprocessing* module, *Model Training* module and *Online Analytics* module.

Data Preprocessing: Both offline data (from both telecommunication company and metro system) and online data (from telecommunication company only) are pre-processed, before sending to model training and online analytics. The major tasks of data preprocessing include: 1) cleaning the data by removing incomplete records from CDR table and unpaired individual records (without either entrance or exist record) from metro data; 2) data aggregation to generate meaningful features for the recurrent neural network model; 3) noise injection to enforce personal privacy preservation.

Model Training: Given the offline data after preprocessing, the model training module runs optimization over recurrent neural network, in order to get the optimal weights for the prediction model. The weights of the result network is forwarded to the online analytics module for real-time prediction. While the training over the models follows the standard strategy on neural network, the configuration of the network is mostly tricky and shows huge impact on the performance. In following subsection, we introduce weight-sharing recurrent neural networks to support co-training on prediction models for multiple metro stations.

Online Processing and Prediction: To handle the massive and highly varying workload of CDR data, we deploy the online processing and analytical system on Resa [1], our in-house elastic streaming analytical platform. The platform runs the analytical algorithm on a cloud and automatically re-scales the resource based on the instantaneous workload.

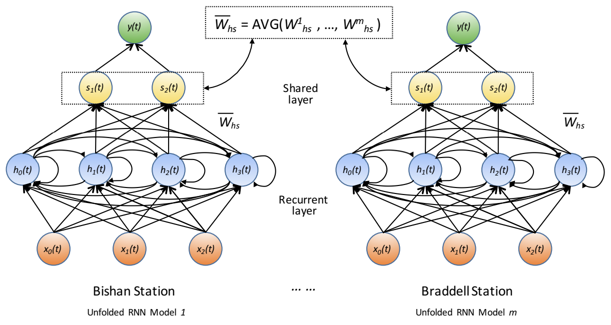


Fig. 2. An example of recurrent neural network with 4 input variables, 3 hidden variables and 1 prediction target.

C. Weight-Sharing Recurrent Neural Networks

Recurrent neural network (RNN) is applicable in discrete time domain, such that a consistent snapshot at each timestamp t always exists. There are three types of variables in RNN, including input variables, hidden variables and target variables. The input variables x are not controllable but time-dependent input from the physical world. The hidden variables h work as memory for the neural network, which maintain key information from past for prediction. The target variables y are the prediction targets of the prediction, e.g. passengers entering metro station at timestamp $t+1$. There is a weight associated with each directed edge in the model. To simplify the notation, the weights on edges from x to h are organized and kept in a matrix W_{hx}. Similarly, matrices W_{hh} (W_{oh} resp.) maintain the weights of self-edges on h (edges from h to y resp.). On each timestamp t, the hidden variables depend on the input x_t as well as the states of the hidden variables in previous timestamp h_{t-1}. Given the weights on the edges, the prediction runs following:

$$\begin{aligned} h_t &= \tanh(W_{hx}x_t + W_{hh}h_{t-1} + b_h) \\ y_t &= \sigma(W_{oh}h_t + b_o) \end{aligned} \tag{1}$$

The training of the model with training data $\{(x_t, y_t\}$ is thus to find the optimal weights in matrices $\{W_{hx}, W_{hh}, W_{oh}\}$ to minimize the prediction error on the known x_ts.

Usually, trajectory of a mobile phone user is a sequence of locations. Considering thousands of users, direct space partitioning by tiles may result in highly skewed distribution of points in the cells. In our system, we apply a more natural feature generation approach by clustering the location based on the *measurement points* of telecommunication company. These points are the locations of the cell towers connected to the nearby mobile phones. The system thus transforms a complicated trajectory to a sequence of measurement points the phone has connected to.

The nodes h in the middle layer of RNN, named recurrent layer, can be designed as different types of units, such as simple node (vanilla), Long-Short Term Memory (LSTM) unit, or Gated Recurrent Unit (GRU). Each unit combines the values of all hidden nodes at previous timestamp h_{t-1} with one input value at the current time x_t. By adding various values recurrently, RNN has the capability to infer the crowd flow in-between any pair of measurement points. Finally, the top layer outputs one value y as the final estimation on the number of passengers entering station in next 15 minutes. The model training is completed by the optimization method of stochastic gradient descent (SGD), which back-propagates the prediction error from the output layer to the input layer.

In our system, we use two strategies to build RNN models over all stations. The intuitive strategy is to build a single RNN model for each metro station. Intuitively, it is possible that some nearby stations may have the similar movement patterns in spatial-temporal aspects. For example, two neighbor stations in CBD area have the same peak hours. In order to include more CDR data from other stations without increasing the computational cost dramatically, we allow several individual RNN models to be trained separately but share a fraction of weights. To accomplish this goal, we add an additional layer between hidden layer and output layer, which is called a *shared layer* shown in Fig. 2. After tuning all weights in back-propagation in each model, the weights on the shared layer across several models are averaged. Similar patterns may be learnt after a number of iterations, and the overall prediction accuracy of all stations can be improved consistently.

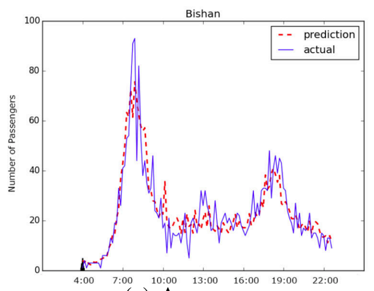
IV. Demonstration Walkabout

In this demonstration, we predict the crowd density of *Bishan* station, an interchange station in Singapore's metro system, with both vanilla-RNN and hyper-RNN (with weight sharing). We also implement a nonlinear autoregressive neural network with external input (NARX) as a strawman approach. The prediction accuracy is calculated by mean absolute percentage error (MAPE). The results show that vanilla-RNN and hyper-RNN are significantly more accurate than NARX, since RNN model has a recurrent layer capturing key information of previous states from different input sources. Moreover, the hyper-RNN performs better than simple vanilla-RNN, because of more inputs from other stations and the weight sharing scheme among models. Fig. 3(a) shows the trends of actual crowd density and prediction of Bishan station in a single day.

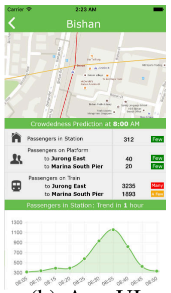
We also demonstrate the usefulness of the technique to cell phone users before their travel in public transportation. Users subscribe and receive crowd density prediction on our own mobile app "Mercury", as shown in Fig. 3(b). The prediction results are updated in real time, with prediction continuously running on our stream analytical engine.

V. Conclusion

In this demonstration, we present a new system *Mercury* which provides prediction on crowd density of metro stations, by analyzing telecommunication signals in nearby region. This is the first investigation on the utilization of teleco data for public transportation and validates the power of recurrent neural network and parallel computation techniques.



(a) Accuracy



Crowdedness Prediction at **8:00** AM

Passengers in Station	312	Few
Passengers on Platform		
to **Jurong East**	40	Few
to **Marina South Pier**	20	Few
Passengers on Train		
to **Jurong East**	3235	Many
to **Marina South Pier**	1893	A Few

(b) App UI

Fig. 3. Example demonstration on user interface and accuracy results.

Acknowledgment

This study is supported by the research grant for the Human-Centered Cyber-physical Systems Programme at the Advanced Digital Sciences Center (ADSC) from Singapores A*STAR. This work is also partially supported by TelePort, a joint project between I2R and ADSC; and the Science and Engineering Research Council (SERC) Grant 152-41-00032.

References

[1] Real-time streaming analytics. https://github.com/ADSC-Cloud/resa.

[2] R. Adhikari and R. K. Agrawal. A novel weighted ensemble technique for time series forecasting. In *PAKDD*, pages 38–49, 2012.

[3] T. Ban, R. Zhang, S. Pang, A. Sarrafzadeh, and D. Inoue. Referential knn regression for financial time series forecasting. In *ICONIP*, pages 601–608, 2013.

[4] G. Biau, K. Bleakley, L. Györfi, and G. Ottucsák. Nonparametric sequential prediction of time series. *Journal of Nonparametric Statistics*, 22(3):297–317, 2010.

[5] G. E. P. Box and G. Jenkins. *Time Series Analysis, Forecasting and Control.* 1990.

[6] J. T. Connor, R. D. Martin, and L. E. Atlas. Recurrent neural networks and robust time series prediction. *Neural Networks, IEEE Transactions on*, 5(2):240–254, 1994.

[7] J. Ding, T. Z. J. Fu, R. T. B. Ma, M. Winslett, Y. Yang, Z. Zhang, and H. Chao. Optimal operator state migration for elastic data stream processing. *CoRR*, abs/1501.03619, 2015.

[8] T. Z. J. Fu, J. Ding, R. T. B. Ma, M. Winslett, Y. Yang, and Z. Zhang. DRS: dynamic resource scheduling for real-time analytics over fast streams. In *ICDCS*, pages 411–420, 2015.

[9] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[10] J. Koutník, K. Greff, F. J. Gomez, and J. Schmidhuber. A clockwork RNN. In *ICML*, pages 1863–1871, 2014.

[11] S. Liu, N. Yang, M. Li, and M. Zhou. A recursive recurrent neural network for statistical machine translation. In *ACL*, pages 1491–1500, 2014.

[12] J. Martens and I. Sutskever. Learning recurrent neural networks with hessian-free optimization. In *ICML*, pages 1033–1040, 2011.

[13] D. Nguyen-Tuong, M. W. Seeger, and J. Peters. Model learning with local gaussian process regression. pages 2015–2034, 2009.

[14] I. Sutskever, J. Martens, and G. E. Hinton. Generating text with recurrent neural networks. In *ICML*, pages 1017–1024, 2011.

[15] U. Thissen, R. Van Brakel, A. De Weijer, W. Melssen, and L. Buydens. Using support vector machines for time series prediction. *Chemometrics and intelligent laboratory systems*, 69(1):35–49, 2003.

[16] M. K. Titsias. Variational learning of inducing variables in sparse gaussian processes. In *AISTATS*, pages 567–574, 2009.

[17] Y. Wang and B. Chaib-draa. A knn based kalman filter gaussian process regression. In *IJCAI*, 2013.

[18] R. J. Williams. Training recurrent networks using the extended kalman filter. In *IJCNN*, pages 241–246, 1992.

[19] J. Zhou and A. K. H. Tung. SMiLer: A semi-lazy time series prediction system for sensors. In *SIGMOD Conference*, pages 1871–1886, 2015.