A Parallel Patient Treatment Time Prediction Algorithm and Its Applications in Hospital Queuing-Recommendation

Popat Borse[1]Samruddhi Raut[2], Amruta Shingade[3], Prachi Girolla[4]

[1]*Department of Computer Engineering*
[2]*Department of Computer Engineering*
[3]*Department of Computer Engineering*
[4] *Department of Computer Engineering*

Abstract* — *Effective patient queue management to cut back patient wait delays and patient overcrowding is one in all the most challenges featured by hospitals. In essential and annoying waits for long periods lead to substantial human resource and time wastage and increase the frustration endured by patients. for every patient within the queue, the entire treatment time of all the patients before him is that the time that he should wait. it would it'd be convenient and desirable if the patients might receive the foremost economical treatment organize and understand the predicted waiting time through a mobile application that updates in real time. Therefore, we've a tendency to propose a Patient Treatment Time Prediction (PTTP) algorithmic to predict the waiting time for each treatment task for a patient. We've a tendency to use realistic patient info from various hospitals to get a patient treatment time model for every task. Supported this large-scale, realistic data-set, the treatment time for each patient at intervals the present queue of every task is predicted. Supported the expected waiting time, a Hospital Queuing Recommendation (HQR) system is developed. HQR calculates Associate in Nursing predicts an economical and convenient treatment set up suggested for the patient. Our proposed model to suggest an efficient treatment set up for patients to minimize their wait times in hospitals.

Keywords- PTTP, HQR, RF

• INTRODUCTION

Patient queue management and wait time prediction kind a difficult and complicated job as a result of every patient would possibly need totally different phases/ operations, like a checkup, numerous tests, e.g., a sugar level or blood test, X-rays or a CT scan, minor surgeries, throughout treatment. Every treatment task will have varied time necessities for every patient, which makes time prediction and recommendation extremely difficult. A number of the tasks are independent, whereas others may need to attend for the completion of dependent tasks. Most patients should wait for unpredictable however long periods in queues, waiting for their turn to accomplish every treatment task.

In this paper, we have a tendency to concentrate on serving to patients complete their treatment tasks in a very predictable time and helping hospitals schedule every treatment task queue and avoid overcrowded and ineffective queues. We use massive realistic information from numerous hospitals to develop a patient treatment time consumption model. The realistic patient information are analyzed carefully and strictly supported important parameters, like patient treatment begin time, end time, patient age, and detail treatment content for every totally different task. we have a tendency to

determine and calculate totally different waiting times for various patients based on their conditions and operations performed throughout treatment.

• PROBLEM STATEMENT

Prediction based on analysis and process of massive noisy patient information from varied hospitals could be a difficult task .Most of the information in hospitals are massive, unstructured, and high dimensional. Hospitals that contain an excellent deal of information, like patient data, medical activity data, time, treatment department, and detailed information of the treatment task. Because of the manual operation and varied unexpected events throughout treatments, a large quantity of incomplete or inconsistent information seems, like a lack of patient gender and age data, time inconsistencies caused by the time zone settings of medical machines from completely different manufacturers, and treatment records with only a start time but no end time.

• LITERATURE REVIEW

The primary paper [1], a substitute rule for modern construction of binary regression bushes is conferred. This rule, known as SAIRT, adapts the elicited version as soon as going through data streams concerning unknown dynamics, like gradual and abrupt function waft, modifications in positive areas of the operate, noise, and digital drift. It conjointly handles every symbolic and numeric attributes. The projected rule will routinely adapt its inner parameters and version structure to get new styles, reckoning on this dynamics of the data movement. SAIRT will display the exceptional of nodes and may neglect examples from unique areas, storing the ultimate ones in local home windows associated to the leaves of the tree. On these conditions, current regression approaches would like a cautious configuration depending on the dynamics of the problem. Experimentation shows that the projected rule obtains better outcomes than present day algorithms as soon as managing information streams that involve changes with absolutely distinct speeds, noise levels, sampling distribution of examples, and partial or whole adjustments of the underlying perform.

the second one paper [2], Gradient Boosted Regression timber (GBRT) is that the present day modern learning paradigm for machine found out net search ranking — a website ill-famed for extremely giant records sets. at some point of this paper, we will be predisposed to propose a unique technique for parallelizing the education of GBRT. Our approach parallelizes the improvement of the man or woman regression trees and operates mistreatment the grasp-employee paradigm as follows. The information is split most of the employees. At each new release, the worker summarizes its statistics-partition mistreatment histograms. The grasp processor makes use of these to create one layer of a regression tree, so send this deposit to the personnel, allowing the employees to create histograms for consecutive layer. Our rule rigorously orchestrates overlap between communiqué and computation to understand clever performance. Since this method relies on records partitioning, and wishes a tiny low amount of verbal exchange, it generalizes to disbursed and shared memory machines, in addition as clouds. We have a propensity to present experimental effects on every shared memory machines and clusters for two massive scale net search ranking facts sets. We have a tendency to illustrate that the loss in accuracy elicited as a result of the bar chart approximation in the regression tree creation will be remunerated for through slightly deeper trees. As an end result, we have a tendency to look no essential loss in accuracy on the Yahoo information units and a virtually tiny reduction in accuracy for the Microsoft LETOR records. Additionally, on shared memory machines, we have a tendency to get virtually remarkable linear velocity-up with as much as regarding forty eight cores at the large facts

sets. On distributed reminiscence machines, we will be inclined to get a hurrying of twenty five with thirty two processors. on account of statistics partitioning our technique will scale to even larger data units, on that one will pretty expect even better speedups.

SR.NO	YEAR	PAPER NAME	AUTHORS	DESCRIPTION
	IEEE Trans,aug 2011	Self-adaptive induction of regression trees	R. Fidalgo-Merino and M. Nunez	The proposed algorithm can automatically adapt its internal parameters and model structure to obtain new patterns, depending on the current dynamics of the data stream.
	World Wide Web (WWW), 2012	Parallel boosted regression trees for Web search ranking	S. Tyree, K. Q. Weinberger, K. Agrawal, and J. Paykin	This approach is based on data partitioning, and requires a small amount of communication, it generalizes to distributed and shared memory machines, as well as clouds. We present experimental results on both shared memory machines and clusters for two large scale web search ranking data sets.

- **ALGORITHM**

DSAR Algorithm

Input:

STrain: the training datasets;

k: the number of CART trees in the HQR model.

Output:

PTTP: The HQR system model based on PTTP algorithm

Step1: **for** i <- 1 to k **do**

Step 2: create training subset **strain** <- sampling(**STrain**);

Step 3: create OOB subset **sOOBi** <-(**STrain - straini**);

Step 4: create an empty **CART** tree hi;

Step 5: **for** each independent variable ***yj*** in ***straini*** do

Step 6: calculate candidate split points **vs <- yj;**

Step 7: **for** each ***vp*** in ***vs*** **do**

Step 8: calculate the best split point **(yj,vp)** <- arg min$\sum$ **xεR**$_{L\ (yi - cL)2+}$ $\sum$**xεR**$_{R\ (yi - cR)2}$**];**

Step 9: end **for**

Step 10: append node **Node(yj,vp)** to hi;

Step 11: split data for left branch $\mathbf{R_{L(yj,vp)}}$ **<- {x|yj ≤ vp}**;

Step 12: split data for right branch $\mathbf{R_{R(yj,vp)}}$ **<- {xjyj > vp}**;

Step 13: **for** each data R in **{RL(yj;vp); RR(yj;vp)} do**

Step 14: calculate 8(vpLjyj) maxi 8(vijy);

Step 15: **if (ϕ(vp$_{(L|R)}$|yj) ≥ϕ(vp|yj)) then**

Step 16: append subnode Node**(yj,vp$_{(L|R)}$) to Node(yj,vp)** as multi-branch;

Step 17: split data to two **forks $\mathbf{R_L}$(yj,vp$_L$) and $\mathbf{R_R}$(yj,vp$_R$);**

Step 18: else

Step 19: collect cleaned data for leaf node **Dleaf <- (IL ≤ yj ≤OL);**

Step20: calculate mean value of leaf node c **1/K $\sum$D$_{leaf}$;**

Step21: end **if**

Step22: end **for**

Step23: remove ***yj*** from ***straini***;

Step24: end **for**

Step25: calculate **accuracy CAi <-I (hi(x)=y)/I (hi(x)=y)+$\sum$I (hi(x)=z) for hi by testing sOOBi**;

Step26: end **for**

Step27: PTTP **H(X,øj) <-1/k$\sum_i^K$ <- [CAi** × **hi**];

Step28**: return** PTTP.

B. BLOCK DEIAGRAM OF SYSTEM

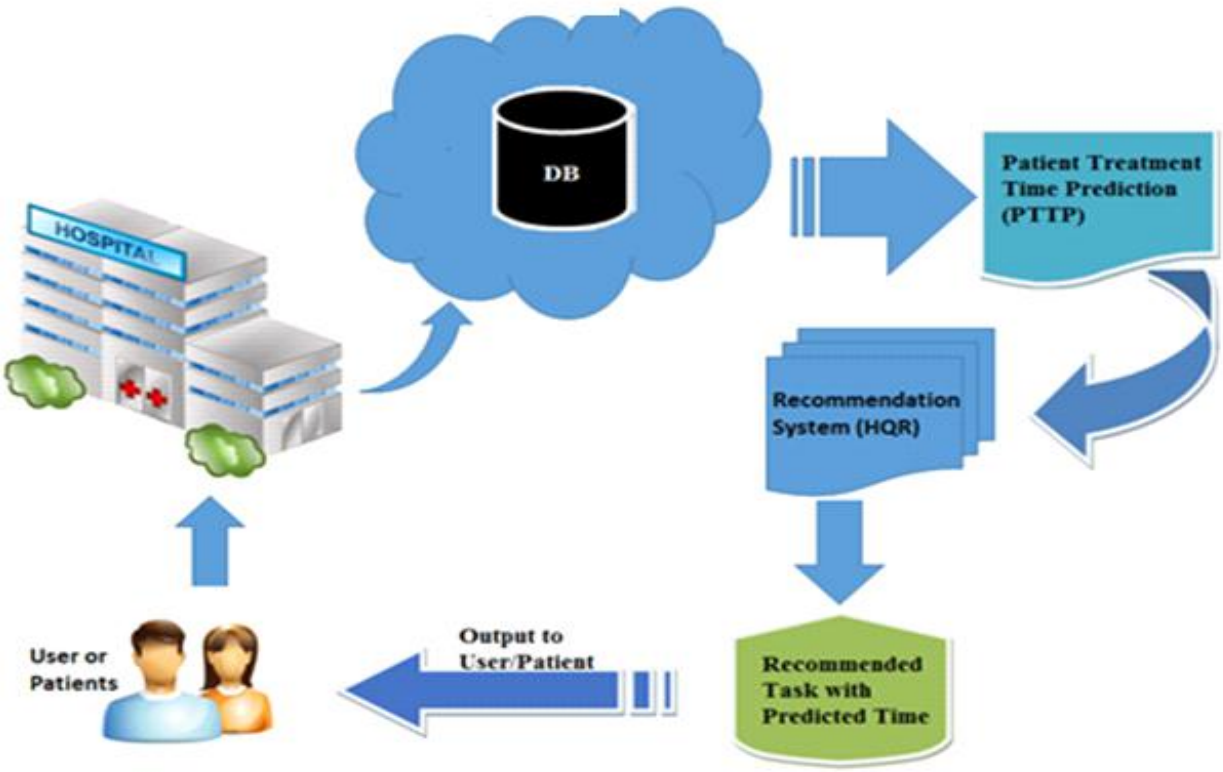

Figure 4.1.Block diagram of Data Stream Analysis

In this paper, a Patient Treatment Time Prediction (PTTP) model is trained based on hospitals' historical data. The waiting time of each treatment task is predicted by PTTP, which is the sum of all patients' waiting times in the current queue. Then, according to each patient's requested treatment tasks, a Hospital Queuing-Recommendation (HQR) system recommends an efficient and convenient treatment plan with the least waiting time for the patient.

- **HARDWARE REQUIREMENT**

SYSTEM	: Pentium IV 2.4 GHz
HARD DISK	: 40 GB
FLOPPY DRIVE	: 1.44 MB
MONITOR	: 15 VGA color

- **SOFTWARE REQUIREMENTS**

- **Operating System**

Windows XP Professional

Java

Code is written in Java. The recommended Java version is JDK 1.6 release and the recommended minimum revision is 31 (v 1.6.31).

Backend
MySQL database

• ADVANTAGES

- Decrease the patients waiting time.
- In this technique, we've a tendency to specialize in helping patients complete their treatment tasks throughout a predictable time and helping hospitals schedule each treatment task queue and avoid overcrowded and ineffective queues.
- To improve the accuracy of the information analysis with continuous options, varied improvement strategies of classification and regression algorithms are proposed.

• APPLICATION

- We will use this system in bus stations for tickets
- ATM's to predict the waiting time to user.
- For railway reservation system

• CONCLUSION AND FUTURE SCOPE

In this project, a PTTP algorithm supported large info and so the Apache Spark cloud environment is planned. A random forest optimization algorithm is performed for the PTTP model. The queue waiting time of each treatment task is expected supported the trained PTTP model. A parallel HQR system is developed, and a cheap and convenient treatment plan is sometimes suggested for each patient. Intensive experiments and application results show that our PTTP algorithmic rule and HQR system succeed high preciseness and performance.

ACKNOWLEDGMENT

Authors wish to acknowledge Principal, Head of department and guide of their project for all the support and help rendered. to express profound feeling of appreciation to their regarded guardians for giving the motivation needed to the finishing of paper.

• REFERENCES

[1] K. Singh, S. C. Guntuku, A. Thakur, and C. Hota, ``Big data analytics framework for peer-to-peer botnet detection using random forests," *Inf. Sci.*, vol. 278, pp. 488497, Sep. 2014.

[2] S. Meng, W. Dou, X. Zhang, and J. Chen, ``KASR: A keyword-aware service recommendation method on MapReduce for big data applications,''IEEE *Trans. Parallel Distrib. Syst.*, vol. 25, no. 12, pp. 32213231, Dec. 2014.

[3] S. Tyree, K. Q. Weinberger, K. Agrawal, and J. Paykin, ``Parallel boosted regression trees for Web search ranking," in *Proc. 20th Int. Conf. World Wide Web (WWW)*, 2012, pp. 387396.

[4] R. Fidalgo-Merino and M. Nunez, ``Self-adaptive induction of regression trees," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 16591672, Aug. 2011.

[5] G. Yu, N. A. Goussies, J. Yuan, and Z. Liu, ``Fast action detection via discriminative random forest voting and top-K sub volume search," *IEEE Trans. Multimedia*, vol. 13, no. 3, pp. 507517, Jun. 2011.

[6] N. Salehi-Moghaddami, H. S. Yazdi, and H. Poostchi, ``Correlation based splitting criterion in multi branch decision tree," *Central Eur. J. Comput. Sci.*, vol. 1, no. 2, pp. 205_220, Jun. 2011.

[7] G. Chrysos, P. Dagritzikos, I. Papaefstathiou, and A. Dollas, ``HC-CART: A parallel system implementation of data mining classi_cation and regression tree (CART) algorithm on a multi-FPGA system," *ACM Trans. Archit. Code Optim.*, vol. 9, no. 4, pp. 47:1_47:25, Jan. 2013.

[8] C. Lindner, P. A. Bromiley, M. C. Ionita, and T. F. Cootes, ``Robust and accurate shape model matching using random forest regression-voting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1862_1874, Sep. 2015.

[9] N. T. Van Uyen and T. C. Chung, ``A new framework for distributed boosting algorithm," in *Proc. Future Generat. Commun. Netw. (FGCN)*, Dec. 2007, pp. 420_423.

[10] Y. Ben-Haim and E. Tom-Tov, ``A streaming parallel decision tree algorithm," *J. Mach. Learn. Res.*, vol. 11, no. 1, pp. 849_872, Oct. 2010.

[11] L. Breiman, ``Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5_32, Oct. 2001.

[12]A Parallel Random Forest Algorithm for Big Data in a Spark Cloud Computing Environment Jianguo Chen, Kenli Li, Senior Member, IEEE, Zhuo Tang, Member, IEEE, Kashif Bilal, Shui Yu, Member, IEEE, Chuliang Weng, Member, IEEE, and Keqin Li, Fellow, IEEE, 1045-9219 (c) 2016 IEEE.

