International Journal of Engineering Applied Sciences and Technology, 2022
Vol. 7, Issue 2, ISSN No. 2455-2143, Pages 423-425
Published Online June 2022 in IJEAST (http://www.ijeast.com)

APPLYING COMPUTATIONAL INTELLIGENCE ON PRODUCTION SUPPORT SOFTWARE MODEL

P.C.Harish Padmanaban
Software Reliability Engineer
Independent Researcher, Bangalore, Karnataka, India

Dr.Yogesh Kumar Sharma
Professor, Software Engineering Dept.
JJT University, Churela, Rajasthan, India

Abstract: **The production support software models are very unpredictable in the field of Information Technology field and people finding more complex ways to solve and mitigate the issues. The normal triage process in any support models starts from identifying the problem and complexity of the assigned production issue. Then the later stage goes on by engaging the right people to fix the issue and updating the Production ticket. Already there are plenty of automation tools are available to minimize the human cost in the field of IT in addition to that this paper providing more insights on production support model and how a approach to minimize the triage process involved in the same model. This paper provides more clarity on existing models and how the new approach will overcome the existing limitations.**

***Keywords*- IT – Information Technology, Incidents, Tickets, POC.**

I . INTRODUCTION:

Production Support is very vast and dry area in the IT industry which in terms composed of more process and software methodologies to fix and update the issues. Usually the issue which impacts the customer who will be reported to the respective point of contact in the organization who delivers the service to the end users. Then the POC work on the incident and find the root cause to analyze the issue. There are several factors to find the root cause and POC's duty to map all of the factors to identify the solution. The important factor as part of the triaging include fixing the issue before the given dead line. The research has been done in various scenarios to identify where and how best we can fit the intelligence system to avoid escalations . And the same research has been applied in several ticketing systems across the technologies.
A Typical triage process includes as below -

Creating Incident → Assigning to the right POC → Root cause analysis → fixing the issue→Update Incident
The above flow is the normal structure followed across all the organizations only the tools and technologies differs in terms of organization's culture but the objective is same.

II. FACTORS WHEN THE PRODUCTION ISSUE AFFECTING THE CUSTOMER WORK FLOW:

1. Recent changes in the code which is not been reviewed properly.
2. Thread / Heap dumps – Improper validation checks to increase the thread size.
3. Permission issue.
4. Installation/build failure.
5. Network issues.
6. Server issues (Web Server/Application server).
7. Improper validation checks.

This 7 factors are more frequently occurring patterns and would include investing more cost and time to resolve all of the issues but the root cause always will be very simple and can be easily avoidable.

III. IDENTIFYING ERROR PATTERS WITH INTELLIGENT MODELS:

With the help of computational intelligence we can mitigate the cost and time by identifying common patterns across all the scenarios with the prediction coverage of >91% and can be avoid the upcoming complex factors by make the system evolving by its own data. This is the main objective of this paper and how best the issues can be learned and fixed by the system. The system here meant to be a scheduler which applies the computational intelligence technique to learn the triage process and the learned data will be saved as a prediction model and will be kept under Knowledge base and will be used whenever it requires to apply the model.

International Journal of Engineering Applied Sciences and Technology, 2022
Vol. 7, Issue 2, ISSN No. 2455-2143, Pages 423-425
Published Online June 2022 in IJEAST (http://www.ijeast.com)

The above techniques are applied with the help of a algorithm called "Machine Learning" and identifying the patterns are implemented using python and R language. The same techniques are tried across different organization culture and the result we received which outperformed all the triage process.

We identified the graph which shows more improvement over time and cost across most of the triage process.

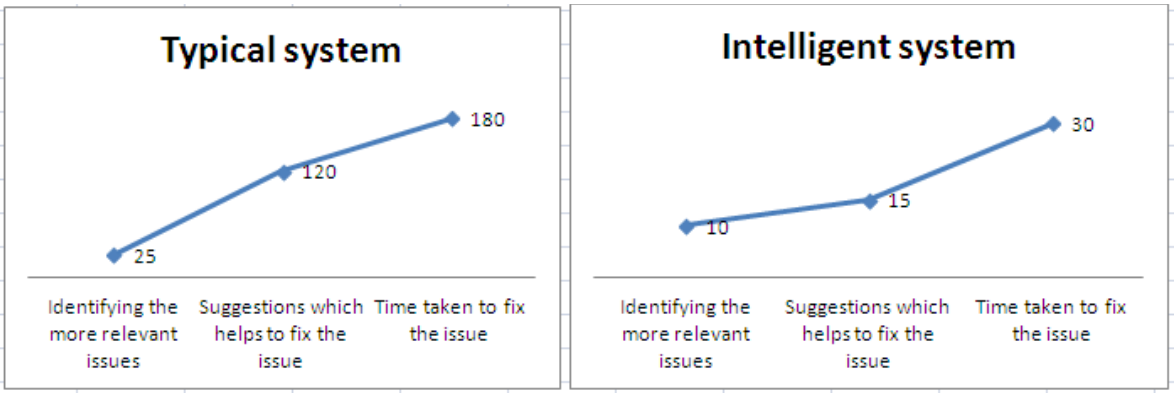

Fig 1.1 – Comparison graph between typical and intelligent systems

IV. IN DETAIL OF THE PROPOSED MODEL:

We named the proposed model as "I-Scheduler" (Intelligent Scheduler) which is a mixture of two intelligence techniques like "learning" and "neural network processing". In this paper we providing the one technique where it is used to solve the production incident and how best it provided the suggestions and which part of the triage process it was helpful.

First the I-Scheduler takes the incident as an input and start processing its intelligent techniques with the provided the data on the incidents. In most of the incidents following points are the key data which in takes acts as an learning data for the I-Scheduler.

- Incident severity & Priority
- Which application and functionality affects the end user
- Due date to fix the issue
- Remedy and relevant issues

The above points are taken as an learning data and with the help of compiler a small search module has been implemented as part of the I-Scheduler which is used to search all the relevant issues and identify more suitable and near coherent pattern with the help of a python library called "deepy" which classifies the relevant patterns and logics are written in such a way where connected dotted outputs are considered as a relevant issue types . The logic includes model to find a more relevant data.

Secondly a Jenkins job has been triggered on the scheduled incidents . Whenever a incident related mail been fall under our SFTP mail client, Then the job will be triggered. The build script has been implemented in shell and run this python scheduler whenever the incidents is been reported.

Finally with the help of a another python library called "Lasagne" we identifying the fix provided by various issues during the initial search phase and providing the best fix suggestions based on the identified relevant patterns.

V. CONCLUSION:

With the default parameters, the code runs for 100 pre-training epochs with mini-batches of size 10. This corresponds to performing 500,000 unsupervised parameter updates. We use an unsupervised learning rate of 0.01, with a supervised learning rate of 0.1. With early-stopping, this configuration achieved a minimal validation error of 1.27 with corresponding test error of 1.34 after 46 supervised epochs.

On an Intel(R) Xeon(R) CPU X5560 running at 2.80GHz, using a multi-threaded MKL library (running on 4 cores), pre training took 615 minutes with an average of 2.05 mins/(layer * epoch). Fine-tuning took only 101 minutes or approximately 2.20 mins/epoch.

Hyper-parameters were selected by optimizing on the validation error. We tested unsupervised learning rates in $\{10^{-1}, \ldots, 10^{-5}\}$ and supervised learning rates in $\{10^{-1}, \ldots, 10^{-4}\}$. We did not use any form of regularization besides early-stopping, nor did we optimize over the number of pre training updates.

VI. REFERENCES:

[1]. Pratik,Jyoshi (2017)`. Artificial Intelligence with Python, (pp.150 – 198).

[2]. Chollet,Francois(2017) . Deep Learning with Python,(pp. 122 – 148).

[3]. WAN,Jiangping(2009) . Research on Software Production Support Structure , "School of Business Administration, South China University of Technology, Guangzhou, China; 2Institute of Emerging Industrializa-tion Development South

China Univ. of Tech., Guangzhou, China"(pp.174-188)

[4]. Poole,David; Alan Mackworth; Randy Goebel(1998). Computational Intelligence A Logical Approach Published by Oxford University Press, Inc.

[5]. Rodolfo C.Cavalcanteab Rodrigo C.Brasileirob; Victor L.F.Souzab ;Jarley P.Nobregab; Adriano L.I.Oliveira(2016) . Computational Intelligence and Financial Markets: A Survey and Future Directions(pp. 195-209)

[6]. Thomas,Hanne ; Rolf,Dornberger (2016) Computational Intelligence in Logistics and Supply Chain Management, Part of the International Series in Operations Research & Management Science book series (ISOR,volume 244)(pp 13-41)

[7]. ShellyXiaonan, Wu; Wolfgang ,Banzhaf(2010).The use of computational intelligence in intrusion detection systems: A review(pp 2-33)

[8]. Dr. Jennifer S. Raj(2019). A COMPREHENSIVE SURVEY ON THE COMPUTATIONAL INTELLIGENCE TECHNIQUES AND ITS APPLICATIONS(pp-147-159)

[9]. Alatas, Bilal(2019) "Sports inspired computational intelligence algorithms for global optimization." ArtificialIntelligence Review 52, no.(pp 1579-1627)

[10]. Ansari, A. Q.(1998) "The basics of fuzzy logic: A tutorial review." COMPUTER EDUCATION-STAFFORD-COMPUTER EDUCATION GROUP-88(pp 5-8)

[11]. Siddique, Nazmul, and Hojjat Adeli(2013) Computational intelligence: synergies of fuzzy logic, neural networksand evolutionary computing. John Wiley & Sons.

[12]. Padmanaban,Harish(2016) DEVELOP SOFTWARE IDE INCORPORATING WITH ARTIFICIAL INTELLIGENCE , VOLUME 1, ISSUE 10 (pp 102-106)

[13]. . M. Poulding and J. A. Clark(2010) "Efficient software verification: Statistical testing using automated search," IEEE Transactions on Software Engineering, vol.36, no. 6, (pp 763–777)

[14]. Oliver Kramer(2008) Self-Adaptive Heuristics for Evolutionary Computation, Vol. 147. ISBN 978-3-540-69280-5.

[15]. T. Miki and T. Yamakawa(1999) Analog implementation of neo-fuzzy neuron and its on-board learning. Computational Intelligence and Application (pp 144–149).

[16]. R. Andrews, J. Diederich and A. B. Tickle(1995) "A survey and critique of techniques for extracting rules from trained artificial neural networks", Knowl.-Based Syst., vol. 8, (pp 373-389)