International Journal of Research Studies in Computer Science and Engineering (IJRSCSE)
Volume 7, Issue 2, 2020, PP 11-16
ISSN 2349-4840 (Print) & ISSN 2349-4859 (Online)
DOI: http://doi.org/10.20431/2349-4859.0702003
www.arcjournals.org

Predicting the Time between failures of Software Projects Using Novel Machine Intelligence Algorithms

Nanwin, Domaka Nuka*, Michael Chief James Philip

Department of Computer Science, Ignatius Ajuru University of Education, Rivers State, Nigeria

Corresponding Author:** ***Nanwin, Domaka Nuka, *Department of Computer Science, Ignatius Ajuru University of Education, Rivers State, Nigeria*

Abstract: *In this paper, machine intelligence technique is used for software effort estimation considering failure patterns and time boundaries in simulation time. Experiments are performed using publicly available software failure dataset and considering two neural machine intelligence techniques, the Hierarchical Temporal Memory (HTM) and the Auditory Machine Intelligence (AMI). The results show that the HTM is a better technique with an error improvement of over 4 times that of the AMI. However, its complexity may hamper its applicability in real time scenarios.*

Keywords: *Failure, Machine Intelligence, Neural, Software Project.*

1. Introduction

Software projects represent complex processes that are initiated through several lifecycles towards the realization of software artifact or produce. Since software development is prone to failures, its state sequel to such failures needs to be estimated in advance to expedite remedial action programs. Software failure trends, thus, are very useful in this regard. However, the estimation of the project outcomes or effort trends leading to such failure remains a perennial system level problem as it involves several inter-related and independent processes that greatly influence its process representation. Also, the need to decipher useful information from limited amount of information is another issue that warrants critical studies. Thus, research is very active in the context of software effort estimation trends including the use of statistical or machine learning approaches such as Arithmetic mean (Am) and Laplace factors (Lf) and Artificial Neural Networks (ANNs) trained by the back-propagation algorithm (Lyu, 1996, ch.10, ch.17).

In order to realize important features in the software project outcome estimation process, trend analysis through refined data mining experiments are typically conducted. While some approaches have considered the time between failures in the context of Maximum Likelihood Estimates (MLE) for reliability growth testing via cumulative failure mode in the time interval (Akuno et al., 2014; Liu et al., 2015) or t-charts as in (Xie et al., 2002); others have focused on the use of machine learning techniques for prediction of fault prone software faults (CATAL, 2016). In this paper, the trend analysis of software project failure data based on time between failures from a systems perspective is considered and presented here. The primary object of this research is to use machine intelligence techniques to determine in advance the time between failures filtering unnecessary data points in the process.

2. Previous Related Works

Some key related works in this field of software project outcome estimation have been investigated. In Cerpa et al (2016), a questionnaire like approach was used to obtain a set of 4 software project outcome data from software engineering field practitioners and from different companies in Chile. They then evaluated the effectiveness of six families of classification methods (Statistical, Nearest Neighbors, Neural Networks, Support Vector Machine (SVM), Decision trees (DTs) and Ensembles) on the obtained field data. From their analysis, they found out that on an integer rank scale of 1-11, the Ensembles called Random Forests (RFs) was the better performing classifier with a rank of No. 1, when compared to other sub-classifiers of the various families. In Kaur & Kaur (2018), six best performing machine learning (ML) algorithms were compared for fault prediction on open source software projects. In their study, they considered standard numeric-based performance metrics, graphical metrics and non-parametric statistical tests for comparative studies. Open source datasets: PMD, EMMA, Find Bugs, Trove and

Dr Java were analysed using the ML models. Their results showed that the Random Forest ML technique is best. This followed by Bagging and Naive Bayesian techniques.

Othman et al (2018) provided insights into the detection of early warning signs prior to failure of a project. They reviewed several existing works methodologies related to early warning signs. Their primary objectives focused on determining at what stage of project life cycle early warning signals of project failure is detected, methodologies used by researchers, and the case studies conducted in the times past. As a primary methodology, they used ad-hoc queries (described by well defined key words) on databases and search engines of well established publishing houses. From their findings, the planning stage was found to be ideal for detecting early warning signs.

Lin et al proposed a graphical model (GM) for software defect prediction. In this GM approach, a monotonically increasing failure time and an estimated two-parameter exponential distribution is used for defect prediction. Failure intensity functions were used for software reliability estimates. Defect data used in experimental analysis showed that GM show improved relative gain when compared to other traditional models.

In a prior study, Lehtinen et al (2014) categorized various causes of software project failures as People, Methods, Tasks and Environment (see Fig.1). They identified the need to study the causal relationships between software processes in order to circumvent consequent failure of software projects. They proposed a unidirectional model (Fig.2) that describes the software process cause-effect relationship across local and bridge levels. Their results indicate that case defect was highly focused on management, implementation and software testing. Benaddy (2018) compared the Recurrent Neural Networks (RNN) with conventional Artificial Neural Networks (ANN) of the Feedforward category for software failure prediction. A real coded genetic algorithm (RCGA) was used to train the feedforward ANN and Adam optimizer used to improve the resilience of the RNN. Their results indicate that the RNN generalize better across the considered datasets.

3. Proposed Methodology

In this section, the Hierarchical Temporal Memory neural technique which is based on the Cortical Learning Algorithms (CLA) proposed earlier in (Hawkins et al., 2010) and the Auditory Machine Intelligence (AMI) developed earlier in (Osegi et al., 2018, Osegi and Anireh, 2019) are presented.

3.1. Hierarchical Temporal Memory

Hierarchical Temporal Memory (HTM) is a machine intelligence algorithm that was developed to solve the missing gaps in the existing back propagation trained neural network scheme. It supports more biological and neuroscience principles and is based on the intelligent processing of cortical columns in the midst of thousands of activating synapses. The architecture of the HTM compared to conventional neural model is shown in Fig.1 and the systems view is presented in Fig 2.

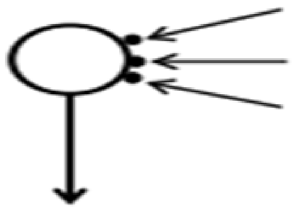

Fig.1a. *Conventional Neural Network Representation*

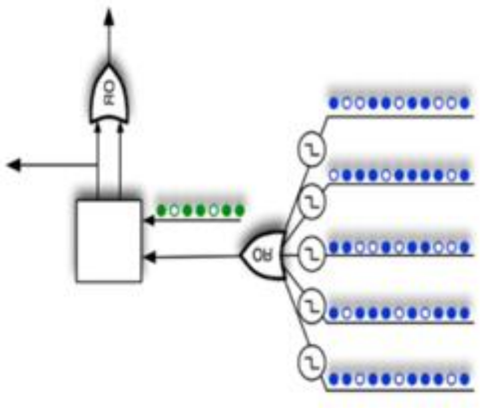

Fig.1b. *HTM Neural Representation (Source: Hawkins et al., 2010)*

As can be seen from Fig.1, the HTM includes a higher set of processing functions that allow added functionality and high-level processing detail when compared to simple neural activation of conventional neural types.

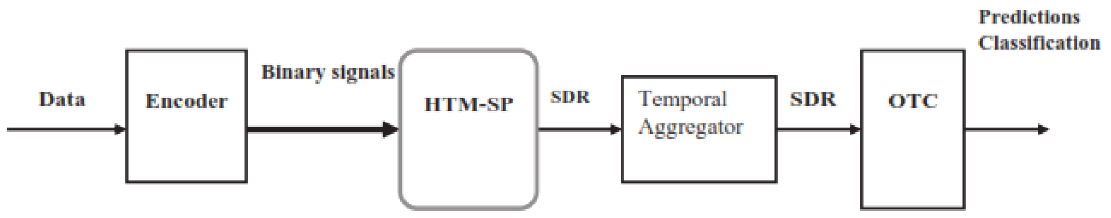

Fig.2. *HTM Systems Level processor (Source: Osegi, 2018)*

In the systems-level process diagram (see Fig.2), the HTM process incoming information in bits using a sparse encoder and a spatial-temporal processing stage. This is then followed by appropriate classifier to generate the output. Indeed, the mathematical formulations of the prediction process in the HTM are complicated and it is beyond the scope of this research paper to treat this matter. Further details of the HTM technique and its mathematical treatment can be found in (Cui et al., 2016, Cui et al., 2017; Osegi, 2018 & Dauletkhanuly et al., 2020).

3.2. Auditory Machine Intelligence

The Auditory Machine Intelligence (AMI) is a deterministic neural machine intelligence technique inspired by the mismatch negativity effect and auditory processing in the mammalian auditory cortex denoted as A1 (Osegi and Anireh, 2019, Osegi et al., 2019, Osegi et al., 2020). It basically comprises of two steps:

Step 1: Low-level prediction for making a prediction in the current time step based on a history of sparse data points in the previous time step. These sparse data points correspond to the evoked potentials originally observed in (Näätänen et al., 1978) as the “odd-ball”.

Step 2: High-level prediction that performs look-ahead predictions several time steps ahead.

In the proposed machine intelligence system for software effort estimation based on failures and time intervals, the Step 1 prediction phase is used. This enables a single (one step ahead) prediction to be made. The AMI-software-failure-prediction-system basically performs this operation using the computation of a single learning as:

$$S_{dev(mean)} = \frac{\left(\left(\frac{\sum [S_{dev}]}{(n-1)}\right) + S_{deviant}\right) - 2}{n+1} \qquad (1)$$

where,

n = number of data points in a temporal sequence

$S_{deviant}$ = the *(n-1)th* value of the temporal sequence

S_{dev} = the difference between $S_{deviant}$ and S_{stars}

S_{stars} = the *(n-2)th* values of the temporal sequence

S^* = sparse set of input sequences

The AMI, then makes a prediction as follows:

$$S_{pred} = S_{deviant} + S_{dev(mean)} \qquad (2)$$

where,

$$S_{deviant} = S_n^* - 1 \qquad (3)$$

$$S_{stars} = S_n^* - 2 \qquad (4)$$

Further details of the AMI technique can be found in (Osegi & Anireh, 2019).

4. Experimental Details and Results

The experiments have been performed using data based on the work of Musa (Musa, 1979, see also Lyu, 1984). The dataset comprises failure data corresponding to the operational life of a real software system. The data features are the Failure Number and the Time-to-Failure and there are 197 instances of these features.

The experiments compare the result of the Auditory Machine Intelligence (AMI) technique with the Hierarchical Temporal Memory (HTM) technique; these techniques are described in Section 3. The key default parameters of the both AMI and HTM techniques are given in Tables 1 and 2 respectively; details of these parameters can be found in (Osegi, 2018; Osegi and Anireh, 2019).

Table1. *Key AMI Parameters*

Parameter	Default value
Model Adjustment Threshold, T_h	0.21
Sparsity factor, s	2

Table2. *Key HTM Parameters*

Parameter	Default value
Number of Columns	250
Initial Synaptic Permanence	0.21
Reduct factor	2
Boost factor	100
Synaptic Permanence Increment	0.1
Synaptic Permanence Decrement	0.1
Number of past sequences used as context	2

In Figures 3 and 4, the error response, Mean Absolute Percentage Error (MAPE) of AMI and HTM techniques are presented. Also, the minimum (min), maximum (max) and mean (u_o) MAPE values for both techniques are presented in Tables 3.

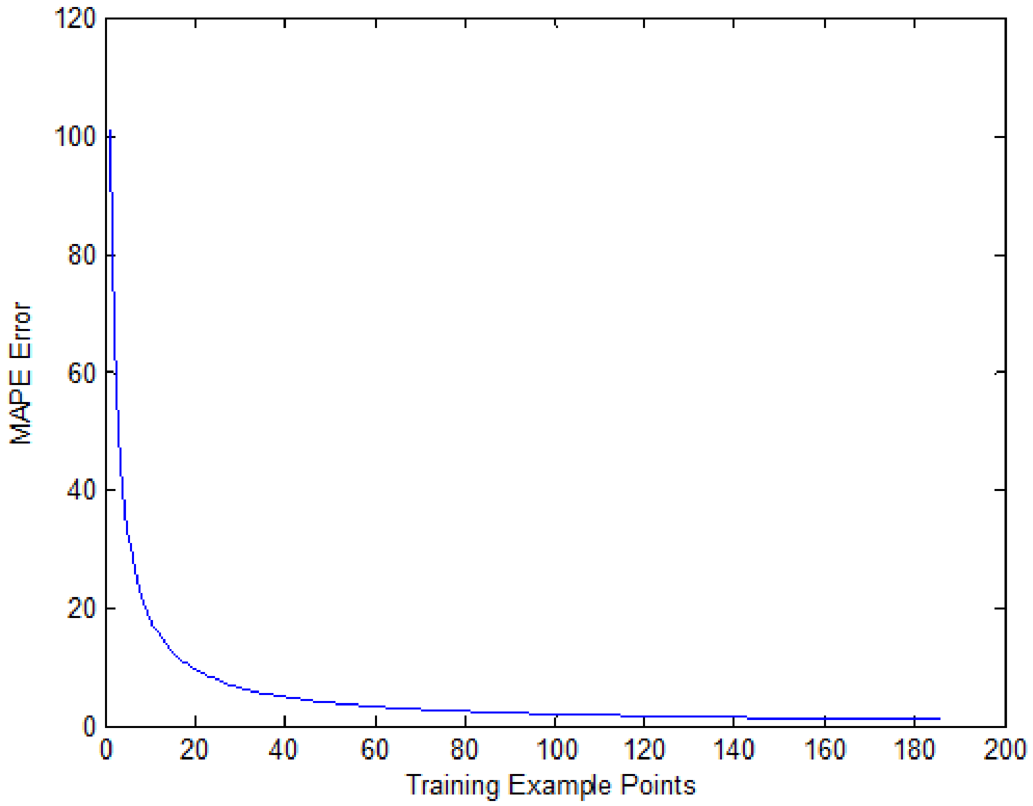

Fig.3. *MAPE response plot of AMI technique for the software failure dataset*

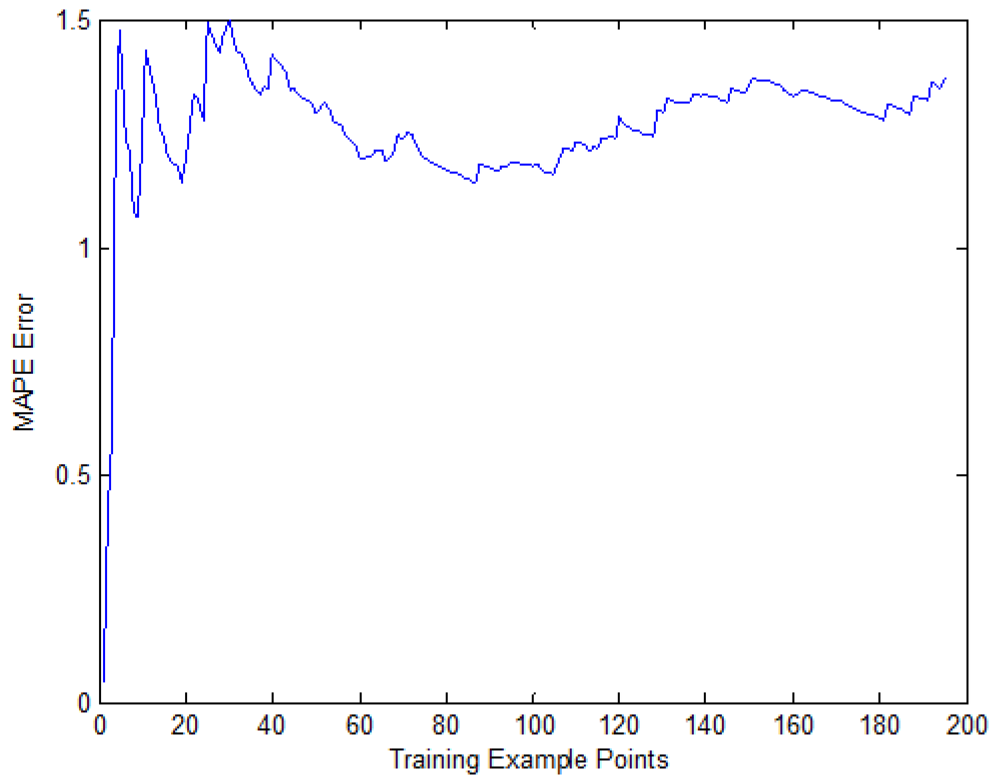

Fig.4. *MAPE response plot of HTM technique for the software failure dataset*

From the Figures (Fig.2-3) it is evident that the HTM performance is better than that of AMI. However, the AMI error response (see Fig.2) is indicative of a progressive improvement through time as the AMI algorithm processes each new data point.

Table3. *AMI Comparative Results*

Error Parameter	HTM$_{MAPE}$	AMI$_{MAPE}$
min	0.05	1.08
max	1.50	100.96
u_o	1.27	5.22

In Table 3, the superiority of the HTM technique over the AMI is clearly obvious with an with an average error improvement of about 4 times that of the AMI.

5. Conclusions and Future Work

This research has presented an experiment on the performance of two machine intelligence algorithms for software effort estimation in the context of predicting the time between failures. The research findings indicate that the Hierarchical Temporal Memory (HTM) is a better technique when compared to the Auditory Machine Intelligence (AMI) technique for the considered dataset. However, AMI neural circuitry is simple to model mathematically and is promising area of research for neural machine intelligence applications. Other areas for future research include investigating more software project failure and effort estimation datasets and refinements to the AMI model to improve its competitive ability.

References

[1] Akuno, A. O., Orawo, L. A. O., & Islam, A. S. (2014). One-Sample Bayesian Predictive Analyses for an Exponential Non-Homogeneous Poisson Process in Software Reliability. *Open Journal of Statistics, 2014*. 4, 402-411

[2] Benaddy, M., El Habil, B., El Meslouhi, O., & Krit, S. D. (2018). Recurrent neural network for software failure prediction. In *Proceedings of the Fourth International Conference on Engineering & MIS 2018* (pp. 1-8).

[3] Çatal, Ç. (2016). The use of cross-company fault data for the software fault prediction problem. *Turkish Journal of Electrical Engineering & Computer Sciences*, *24*(5), 3714-3723.

[4] Cerpa, N., Bardeen, M., Astudillo, C. A., & Verner, J. (2016). Evaluating different families of prediction methods for estimating software project outcomes. *Journal of Systems and Software*, *112*, 48-64.

[5] Cui, Y., Ahmad, S., & Hawkins, J. (2016). Continuous online sequence learning with an unsupervised neural network model. *Neural computation*, *28*(11), 2474-2504.

[6] Cui, Y., Ahmad, S., & Hawkins, J. (2017). The HTM Spatial Pooler—A Neocortical Algorithm for Online Sparse Distributed Coding. *Frontiers in Computational Neuroscience*, *11*.

[7] Dauletkhanuly, Y., Krestinskaya, O., & James, A. P. (2020). HTM theory. In *Deep Learning Classifiers with Memristive Networks* (pp. 169-180). Springer, Cham.

[8] Hawkins, J., Ahmad, S., & Dubinsky, D. (2010). Hierarchical temporal memory including HTM cortical learning algorithms. *Techical report, Numenta, Inc, Palto Alto. https://web.archive.org/web/20110714213347/http://www.numenta.com/htm-overview/education/HTM_CorticalLearningAlgorithms.pdf*.

[9] Kaur, A., & Kaur, I. (2018). An empirical evaluation of classification algorithms for fault prediction in open source projects. *Journal of King Saud University-Computer and Information Sciences*, *30*(1), 2-17.

[10] Lehtinen, T. O., Mäntylä, M. V., Vanhanen, J., Itkonen, J., & Lassenius, C. (2014). Perceived causes of software project failures–An analysis of their relationships. *Information and Software Technology*, *56*(6), 623-643.

[11] Liu, Y., Xie, M., Yang, J., & Zhao, M. (2015, August). A new framework and application of software reliability estimation based on fault detection and correction processes. In *2015 IEEE International Conference on Software Quality, Reliability and Security* (pp. 65-74). IEEE.

[12] Lyu, M. R. (1996). *Handbook of software reliability engineering* (Vol. 222). CA: IEEE computer society press.

[13] Musa, J. D. (1979). Validity of execution-time theory of software reliability. *IEEE Transactions on Reliability*, *28*(3), 181-191.

[14] Näätänen, R., Gaillard, A. W., & Mäntysalo, S. (1978). Early selective-attention effect on evoked potential reinterpreted. *Acta psychologica*, *42*(4), 313-329.

[15] Othman, I., Ghani, S. N., Mohamad, H., Alalou, W., & Shafiq, N. (2018). Early warning signs of project failure. In *MATEC Web of Conferences* (Vol. 203, p. 02008). EDP Sciences.

[16] Osegi, E. N. (2018). Using the hierarchical temporal memory spatial pooler for short-term forecasting of electrical load time series. *Applied Computing and Informatics*.

[17] Osegi, E. N., Anireh, V. I., & Onukwugha, C. G. (2018, June). pCWoT-MOBILE: a collaborative web based platform for real time control in the smart space. iSTEAMS SMART-MIINDs Conference, 13(3), 237-250.

[18] Osegi, E.N., Anireh, V.I.E.: AMI: An Auditory Machine Intelligence Algorithm for Predicting Sensory-Like Data. (2019 in press).

[19] Osegi, E.N., Taylor, O.E., Wokoma, B.A., & Idachaba, A.O. (2019). A Smart Grid Technique for Dynamic Load Prediction in Nigerian Power Distribution Network. International Conference on Sustainable and Innovative Solutions for Current Challenges in Engineering & Technology (ICSISCET- 2019), Gwalior, India.

[20] Osegi, E.N., Taylor, O.E., Wokoma, B.A., & Idachaba, A.O. (2020, in-press). A Smart Grid Technique for Dynamic Load Prediction in Nigerian Power Distribution Network. In Intelligent Computing Applications for Sustainable Real-World Systems. Springer-Nature, Switzerland.

[21] Xie, M., Goh, T. N., & Ranjan, P. (2002). Some effective control chart procedures for reliability monitoring. *Reliability Engineering & System Safety*, *77*(2), 143-150.

Citation: *Nanwin, Domaka Nuka, Michael Chief James Philip, (2020). "Predicting the Time between failures of Software Projects Using Novel Machine Intelligence Algorithms", International Journal of Research Studies in Computer Science and Engineering (IJRSCSE), 7(2), pp.11-16. DOI:http://doi.org/10.20431/2349-4859.0702003*