Contents lists available at Science-Gate

International Journal of Advanced and Applied Sciences

Journal homepage: http://www.science-gate.com/IJAAS.html

Efficient effort estimation of web based projects using neuro-web

Nosheen Qamar [1,*], Farwa Batool [2], Kashif Zafar [2]

[1]Computer Science and Information Technology Department, University of Lahore, Lahore, Pakistan
[2]Computer Science Department, National University of Computer and Emerging Sciences, Lahore, Pakistan

ARTICLE INFO

Article history:
Received 16 May 2018
Received in revised form
29 August 2018
Accepted 2 September 2018

Keywords:
Effort estimation
Neural networks
Software engineering
Web applications
Software planning

ABSTRACT

The effort estimation needs to be done at early stages for successful delivery of software. Numerous models have been developed to estimate software effort during the last decades, but effort estimation of a software project is still a challenging task and in the case of web based projects, it is even harder. The selection of programming language and use of different type of objects i.e. hyperlinks, graphics, and scripts etc. make the web effort estimation process really complex. An estimation model "WebMo", proposed to estimate the effort of web based projects inspired by COCOMO. This research presents a non-algorithmic model named "Neuro-Web" based on Artificial Neural Networks (ANN). The proposed model will use the WebMo parameters as input. These parameters include web application size, productivity coefficients, and 9 different cost drivers. This proposed model is calibrated using the dataset of 164 real-life web applications developed by different freelancers and software houses. The "Neuro-Web" model is compared with the existing model "WebMo" and results reveal that Neuro-Web performs better than "WebMo". The MMRE of the proposed method is just 9.92% as compared to 26.27% for WebMo.

1. Introduction

Effort estimation of a project is a perplexed activity that needs to be done at early stage of the project development. It is the process of forecasting the expected development cost and time. Correct estimation is critically important as under-estimation can results in low quality of the project and eventually leads to the project failure. On the other hand, the over-estimation of a project can be source of business loss (Hill, 2010).

The software effort estimation can be performed by algorithmic (Sharma, 2013) and non-algorithmic methods. The algorithmic methods include COCOMO model (Boehm, 1984), SLIM model (Putnam, 1978) and Function Point based model (Sheta et al., 2008). These methods use different type of parameters. The parameter values are provided to mathematical formulas to foresee software effort. The limitations of these algorithmic methods include its source of estimation (SRS document), inappropriate measurement of project size and difficulty in modeling the complex inherent relationships (Clark et al., 1998). Due to these deficiencies, the researchers focused on non-algorithmic methods based on Computational Intelligence i.e. Genetic Algorithms (Singh and Misra, 2012), Fuzzy Logic (Martin et al., 2005) and Artificial Neural Networks (ANN) (Santani et al., 2014). The ANN has the capability to learn from test data and produce output like a human brain. It can also model the complex relationships between dependent and independent variables effectively (Briand and Wieczorek, 2002).

Along with traditional software, the effort estimation of web based projects is also critically important. The continuously increasing online retail sales ($2,197 trillion in 2017) (Saleh, 2017) reflects the importance of successful web development and accurate effort estimation is the foundation of this success. The demand of being quick-to-market makes web development different from traditional one. The other difference includes its complex nature, small team size, use of multiple techniques (scripts, API's and graphics etc.) (Reifer, 2000) and ad-hoc processes. Despite of these differences, a very few models have been proposed to exclusively deal with web based projects. One of those is WebMo, proposed by Bohem (1984). This model uses the Web Objects metric for effort estimation. The Web Objects include; building blocks, web components, graphics or multimedia files etc. (Reifer, 2000).

* Corresponding Author.
Email Address: nqz786@gmail.com (N. Qamar)
https://doi.org/10.21833/ijaas.2018.11.004

The above mentioned limitations of algorithmic methods attracted us to look for non-algorithmic methods to estimate effort of web based projects. Very few researchers had attempted ANN for estimation web based project. This motivated us to propose a novel non-algorithmic model called Neuro-Web. The model has been verified with the help of 164 real life web based projects.

The remaining paper is structured as follows: Section 2 discusses the Background, Section 3 describes the Literature Review, Section 4 discusses the Proposed Methodology, Section 5 describes the Experimental Setup, Section 6 presents the Results and Analysis and finally the section 7 concludes the paper with some future directions.

2. Background

2.1. WebMo model

Boehm (1984) introduced the WebMo model for effort estimation of web based projects (Reifer, 2000). The WebMo model is derived from COCOMO; a widely used model for effort estimation of traditional projects. The core difference between WebMo and COCOMO is number of cost drivers and the sizing metrics. The COCOMO has 15 cost drivers and WebMo deals with 9 cost drivers. The project size is calculated in Source Lines of Code (SLOC) for COCOMO whereas the WebMo uses the Web Objects metric for size calculation. The Web Objects includes; API's, JavaScript Applet, Graphics, Hyperlinks, application points, components or multimedia files etc. The size of the project is calculated with help of Halstead (1977)'s equation as given below (see Eq. 1). The volume is computed to calculate the size of web project (Halstead, 1977).

$$V^* = Nlog2(n) = (N1^* + N2^*)log2(n1^* + n2^*) \quad (1)$$

where, N is No. of occurrences of web objects and operations on those web objects, n is No. of unique web objects and operations, N1* is total occurrences of web objects, N2* is total occurrences of operations on web objects, n1* is No. of distinct web objects, n2* is No. of distinct operations on web objects, and V* is volume/size of the project.

Eq. 2 is used to calculate the effort (in person-months) of the Web based project.

$$Effort = A\prod_{i=1}^{9} cd_i(Size)^{P1} \quad (2)$$

where, Size is Size as calculated from Halstead's equation, Cd is 9 cost drivers, A is constants, and P1 is power laws.

Table 1 contains the values of A and P1 for different categories of web based projects. Table 2 describes the details of 9 cost drives. The value of cost driver can be measured in an ordinal scale (Very Low, Low, Nominal, High and Very High).

Table 1: Web Development Model Parameter Values (Reifer, 2000)

Domains	A	P1
Web-based electronic	2.3	1.05
Financial/trading applications	2.7	1.05
Business-to-business applications	2.0	1.00
Web-based information utilities	2.1	1.00

**Either 0.5 or 0.33 depending on the scaling (>40 Web Objects)*

Table 2: WebMoEffort Multipliers/Cost Drivers (Reifer, 2000)

Cost Driver	VL	L	N	H	VH
Product Reliability and Complexity (RCPX)	0.63	0.85	1.0	1.30	1.67
Platform Difficulty (PDIF)	0.75	0.87	1.00	1.21	1.41
Personnel Capability (PERS)	1.55	1.35	1.00	0.75	0.58
Personnel Experience (PREX)	1.35	1.19	1.00	0.87	0.71
Facilities (FCIL)	1.35	1.13	1.00	0.85	0.68
Schedule (SCED)	1.35	1.15	1.00	1.05	1.10
Teamwork (TEAM)	1.45	1.31	1.00	1.75	1.62
Process Efficiency (PEFF)	1.35	1.20	1.00	0.85	0.65
Reuse (RUSE)	-	-	1.00	-	-

2.2. Artificial neural networks

Artificial Neural Network (ANN) is a network consists of artificial neurons (Richard, 1987). These neurons are connected to each other with help of connection links as shown in Fig. 1. Some weights are associated with each connection. These weights contain information about input signals which is used to solve some specific problem. In ANN, the nodes are organized in different layers. These layers are consisted of input layer, hidden layer(s) and output layer. The inputs need to be provided to each neuron. Every neuron has its internal state called activation level of neuron. The inputs with aggregated weights, measured on some threshold are provided to activation function to produce the output of that neuron. A large number of activation functions are used depending on nature of problem, like Linear, Sigmoid, Gaussian, Tangent, Hyperbola, Parabola etc.

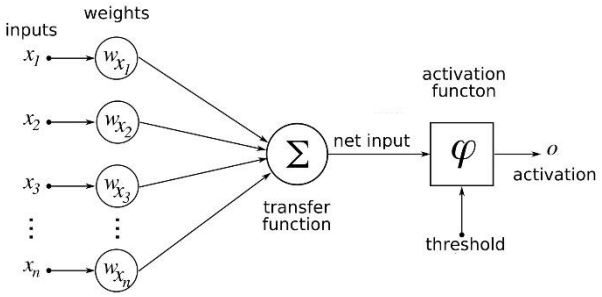

Fig. 1: Artificial neural network basic model

The ANN needs to be trained like human brain. A large number of algorithms exist for training purpose of neural networks, but which algorithm will best works for some artificial neural network,

depends on the architecture of that network. The most widely used topology or architecture of ANN is the feed-forward neural networks (FFNN). The information flows from input neurons to output neurons and never goes in reverse direction in feed-forward network. The intermediate layers called hidden layers can be used to increase the dimensionality of neural network (Richard, 1987).

3. Literature review

One of the important needs of software project management is precise, consistent and accurate prediction of resources. Researchers are working in this direction from last twenty years but still many challenges are associated with cost and effort prediction (Bhatnagar et al., 2010).

The researchers more focus was on minimizing the subjectivity of traditional software estimation methods. A very few researchers took web based project estimation under consideration. There are different algorithmic, regression-based and parametric models like COCOMO model (Boehm, 1984), SLIM's model (Putnam, 1978) and Function Point Analysis (Sheta et al., 2008) and Ordinal Regression Model (Sentas et al., 2005) for traditional software. There are also non-algorithmic techniques exist for software effort estimation. These techniques include; case base reasoning (Mukhopadhyay et al., 1992), clustering (Zhong et al., 2004), artificial neural networks (ANN) (Richard, 1987) and genetic algorithms (GA) (Martin et al., 2005). Due to the successful application of genetic algorithms (Qamar et al., 2018) and artificial neural networks (Richard, 1987) in different domains (i.e. medicine, geology, engineering, image processing, physics, classification and control problems), It grabs the attention of more researchers to use this for software effort estimation and many researchers used this in different areas of software project management.

Tronto et al. (2008) and Bhuyan et al. (2014) evaluated the use of artificial neural networks as prediction of cost and effort in software project management. Furthermore, Finnie et al. (1997) reported that back propagation learning algorithm on multilayer perceptron for software effort prediction. Srinivasan and Fisher (1995) also used multilayer perceptron for effort prediction on COCOMO dataset.

Ruhe et al. (2003) used hybrid techniques for web based projects estimation. He used small dataset from industry. The multivariable regression and expert judgment were the used techniques to estimate effort. Later, Costagliola et al. (2006) performed the comparison between two types of web based measures for size estimation. Mendes (2007) used Bayesian Network for effort estimation and found it better than regression-based model. Mendes (2007), further used Classification and Regression Trees (CART) and case-based reasoning (CBR) techniques for web based project estimation. The WebMo model (Reifer, 2000) proposed by Boehm (1984) was also used to estimate the effort of web projects.

Reddy et al. (2007) proposed an approach for web effort estimation using ANN in 2007. Later on, Panda (2015) used artificial neural networks to estimate effort of Agile and web based projects in 2015. The results showed that ANN performed better than previous techniques. Recently, Aghazadeh and Gharehchopogh (2018) proposed a Hybrid model of Multi-layer Perceptron Artificial Neural Network and Genetic Algorithms in Web Design Management Based on CMS.

4. Proposed methodology

The WebMo is the algorithmic model which is developed for web-based projects. We had rectified this model in Multilayer Artificial Neural Network by providing the parameters of WebMo model to the ANN as input and estimated effort was measured by training the ANN. A detailed comparison between actual efforts, WebMo's estimated and proposed model's (Neuro-Web) effort was conducted.

A Feed Forward Neural Network was designed (as shown in Fig. 2) which is taking 9 cost drivers and calculated size in its input neurons layer and there are five neurons in hidden neuron layer and one output neuron. The number of neurons in hidden layer was selected after an iterative testing process by keeping in view that more neuron can cause the issue of over fitting.

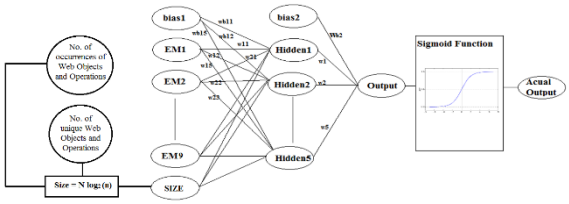

Fig. 2: Architecture of ANN used for neuro-web model

The steps of Neuro-Web Model are given below:

Step 1: Get values against all cost drivers
Step 2: Initialize the weights, biases and number of nodes in hidden layer.wi=whi=1; biasi=1
Step 3: Set learning rate α = 0.003
Step 4: Test stopping condition for false,
Repeat the steps 5 to 12
Step 5: For each training data,
Repeat the steps 6 to 12
*Step 6: Compute the hidden layers Hiddenj = b1 +Σ Xi*wij for i=1 to 16; j = 1 to n*
Step 7: Activate the hidden layers
Hiddeni =1/1+e-H for i = 1 to n (number of hidden nodes)
Step 8: Compute the output layer
*effort = bias2 + Hidden1*wh1+...+Hiddenn*whn*
Step 9: Compute error
error = ln(Actual Effort) – efforts
Step 10: Compute Δw
Step 11: Update the weights using Δw
Step 12: Test stopping condition
Repeat Step 13 and 14 for all projects in test data
Step 13: Compute effort of testing data

Pick weight from weight associative memory of that training project which gives effort closest to actual effort.

Learning rate α = 0.003 and stopping condition is error should be less than some threshold.

Initially, all weights (input and hidden layer) and bias are set on 1. For each training data step 6 to step 12 are perform, in these step weights are being updating and saving in a weight associative memory. In step 6 and 7 weights of every connection of input and hidden layer is computed respectively. In step 8, effort is computed with the help of input and hidden layer weights. From step 9 to 12, weights are being updating while error (actual – estimated effort) is greater than defined threshold. In step 13, estimated effort is computed for training data, for this purpose weight associative memory is used. The accuracy of estimated effort is calculated by using the most popular method such as Magnitude Relative Error (MRE) and Mean Magnitude Relative Error (MMRE) (Briand et al., 1999) which are described in Eq. 3 and Eq. 4.

$$\text{MRE} = \frac{|\text{Actual Effort} - \text{Estimated Effort}|}{\text{Actual Effort}} * 100 \quad (3)$$

$$\text{MMRE} = \frac{1}{\text{N}} \sum_{x=1}^{n} MREi \quad (4)$$

5. Experimental setup

The dataset was collected from different software houses and freelancers to analyze and implement the effort estimation model. The companies provide us information on a condition of hiding their identity and to use this information just for research purpose. The Fig. 3 depicts the details of dataset collected from different sources. Around 40% of the data sets were collected from different software houses of Pakistan, more particularly from Lahore. The other sources of dataset (17%) were the freelancers working in virtual teams for clients. Around 43% of projects data was taken anonymously (developers did not disclosed their affiliation with any company).

The total of 164 projects dataset was divided into training (61%) and testing (39%) parts as Neural Network works in two modes: training and testing mode. During training mode, we used training data to adjust the weights. While the testing mode will validate either network is trained properly or not on provided testing dataset. The single instance of data set (a project) contains project name (project pseudonym), its launching year, values of its 9 cost drivers in term of very low, low, nominal, high and very high, occurrences of operands and operators in a project, distinct operands and operators and its actual effort in person months was given. Each attribute is comma separated. Table 3 demonstrates the values of parameters used to implement the feed forward neural network.

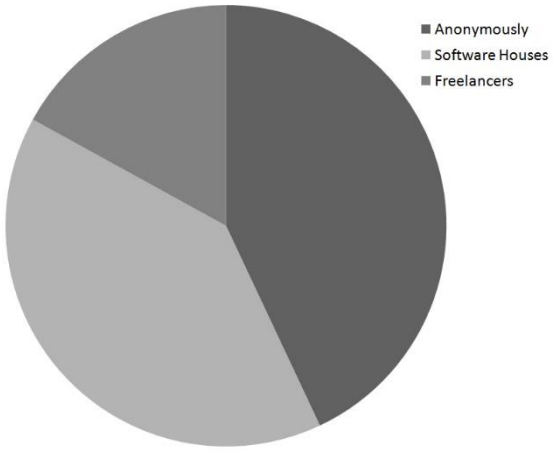

Fig. 3: Sources of dataset collected

Table 3: Experimental Values taken for implementation

Parameters	Values
Convergence Objective	0.01
Learning Rate	0.006
Architecture Used	Feed Forward Neural Network
Training Method Used	Trainlm (Levenberg-Marquardt)
No. of Training Data	100
No. of Testing Data	64
Function Used	Sigmoid

6. Results and analysis

The overall results for 64 projects (Testing Dataset) are presented in Table 4. The second column represents the actual effort taken from software houses, anonymous sources and freelancers with collected dataset. The 3rd and 4th columns compare the estimated effort using Neuro-Web and WebMo respectively. The results show that the Neuro-Web performed much better than WebMo as Neuro-Web estimated effort was much closer to the actual effort. The last two columns of the table calculate the MRE of Neuro-Web and WebMo for estimated effort. It can be seen in table 4 that mean relative error of Neuro-Web is much lesser than WebMo and ultimately the MMRE too. The overall results demonstrate that Neuro-Web performed much better than WebMo. Fig. 4 depicts the comparative analysis between actual efforts, Neuro-Web's estimated effort and estimated effort of WebMo. It can be shown though figure that estimated effort of Neuro-Web is much closer to the actual effort when comparing with WebMo.

7. Conclusion

In this paper, we have proposed a novel non-algorithmic model Neuro-Web for effort estimation of web-based projects. The proposed model is calibrated with help of 164 real life project's dataset. The model is based on artificial neural network that need to be trained like human brain. The model used WebMo model parameters as input. The estimated effort of proposed model was compared with actual effort and the effort measured using WebMo model. The estimated effort using Neuro-Web was close to

actual effort. The MMRE of Neuro-Web was just 9.92% while the MMRE of WebMo was 26.27%.

Table 4: Comparison between actual, neuro-web and WebMo based on effort and MRE

Sr#	Actual Effort	Effort using WebMo	Estimated Effort using Neuro-Web	MRE % using WebMo	MRE % using Neuro-Web
1	51.00	55.76	31.68	9.33	37.88
2	61.00	60.50	61.39	0.83	0.63
3	43.00	48.48	31.68	12.73	26.32
4	61.00	59.58	61.39	2.32	0.63
5	450.00	399.32	439.71	11.26	2.29
6	43.00	5.40	31.68	87.44	26.32
7	115.00	62.65	117.82	45.52	2.45
8	1250.00	1011.53	1235.97	19.08	1.12
9	2450.00	2640.28	2376.81	7.77	2.99
10	1400.00	1394.60	1354.74	0.39	3.23
11	980.00	1343.16	963.57	37.06	1.68
12	410.00	518.45	415.92	26.45	1.44
13	2400.00	2575.87	2376.81	7.33	0.97
14	425.00	587.50	439.71	38.24	3.46
15	255.00	353.59	236.63	38.66	7.20
16	110.00	66.76	117.82	39.31	7.11
17	575.00	519.05	566.45	9.73	1.49
18	105.00	56.48	117.82	46.21	12.21
19	160.00	290.63	168.32	81.64	5.20
20	780.00	545.38	748.62	30.08	4.02
21	2150.00	1389.96	2099.50	35.35	2.35
22	375.00	478.72	366.33	27.66	2.31
23	285.00	381.15	297.03	33.74	4.22
24	9.00	3.59	24.75	60.08	175.03
25	725.00	737.97	742.77	1.79	2.45
26	452.00	501.90	439.71	11.04	2.72
27	2462.00	1422.79	2376.81	42.21	3.46
28	165.00	87.69	168.32	46.85	2.01
29	152.00	106.33	168.32	30.04	10.73
30	640.00	572.58	629.87	10.53	1.58
31	885.00	582.67	873.53	34.16	1.30
32	450.00	648.84	439.71	44.19	2.29
33	195.00	105.60	207.92	45.85	6.63
34	580.00	579.16	566.45	0.15	2.34
35	440.00	567.44	439.71	28.96	0.07
36	73.00	73.73	61.39	0.99	15.91
37	303.00	404.78	297.03	33.59	1.97
38	302.00	391.83	297.03	29.75	1.65
39	242.00	290.02	236.63	19.84	2.22
40	605.00	540.16	594.17	10.72	1.79
41	760.00	589.15	742.77	22.48	2.27
42	1220.00	1116.99	1235.97	8.44	1.31
43	100.00	99.96	117.82	0.04	17.82
44	412.00	494.61	415.92	20.05	0.95
45	4180.00	2504.25	4129.81	40.09	1.20
46	1775.00	1127.78	1755.94	36.46	1.07
47	1650.00	1143.18	1755.94	30.72	6.42
48	1950.00	1066.52	1927.16	45.31	1.17
49	705.00	541.08	713.03	23.25	1.14
50	1350.00	982.52	1354.74	27.22	0.35
51	481.00	474.53	475.38	1.35	1.17
52	600.00	453.25	594.17	24.46	0.97
53	432.00	408.68	439.71	5.40	1.78
54	650.00	284.35	629.87	56.25	3.10
55	480.00	372.26	475.38	22.45	0.96
56	13.00	5.54	31.68	57.39	143.72
57	39.00	36.35	31.68	6.80	18.76
58	1255.00	1471.57	1235.97	17.26	1.52
59	100.00	99.96	117.82	0.04	17.82
60	415.00	497.14	415.92	19.79	0.22
61	85.00	62.11	89.11	26.93	4.83
62	66.00	58.53	61.39	11.31	6.99
63	176.00	73.22	168.32	58.40	4.37
64	651.00	515.83	629.87	20.76	3.25
		% MMRE		**26.27**	**9.92**

As of now, we have taken this dataset from anonymous sources, Pakistani software houses and freelancers only. In future, this model can be evaluated using the datasets taken from different international software houses. Similarly, instead of comparing this model with WebMo, this could be compared with other web effort estimation tools/models. The model can be calibrated by using different company sizes, different areas of web based applications and different number of datasets. Last, but not the least, the research can be extended for different other non-algorithmic techniques like Fuzzy techniques, Swam Intelligence and Genetic Algorithms.

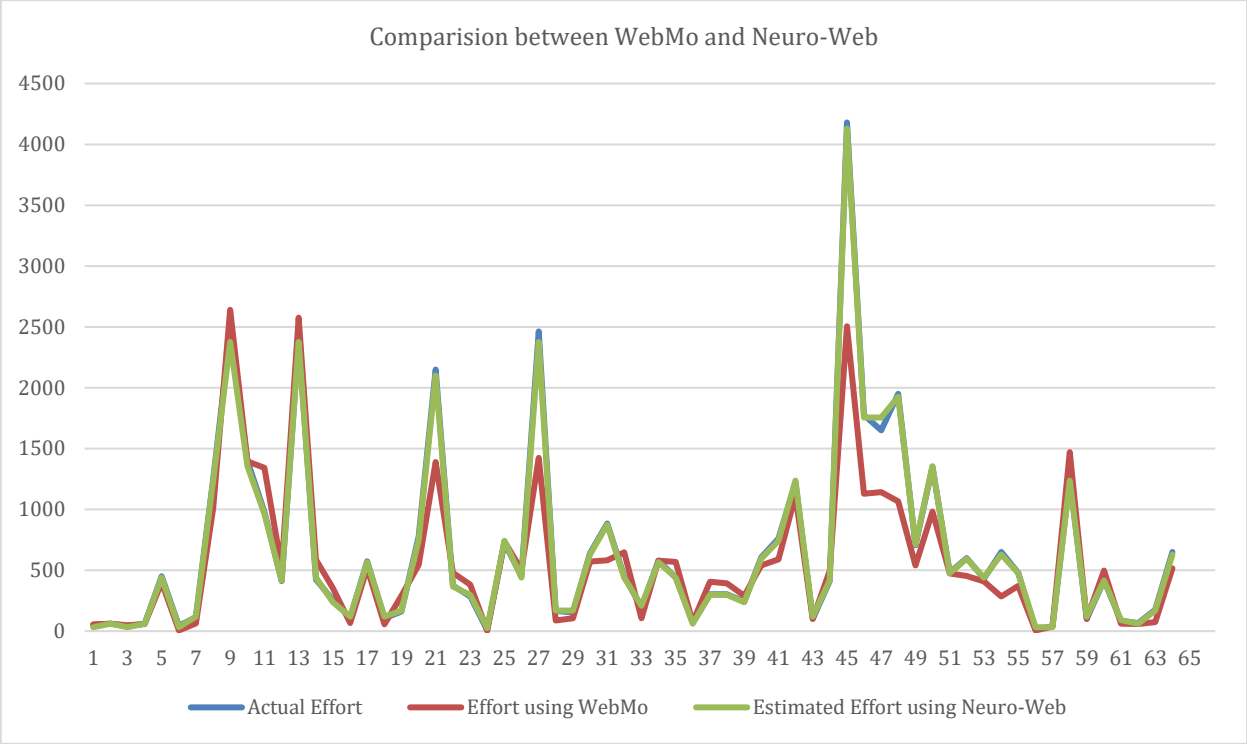

Fig 4: Comparison between actual effort, estimated WebMo effort and neuro-web effort

References

Aghazadeh M and Gharehchopogh SF (2018). A new hybrid model of multi-layer perceptron artificial neural network and genetic algorithms in web design management based on CMS. Journal of AI and Data Mining, 6(2): 409-415.

Bhatnagar R, Bhattacharjee V, and Ghose MK (2010). Software development effort estimation–neural network vs. regression modeling approach. International Journal of Engineering Science and Technology, 2(7): 2950-2956.

Bhuyan MK, Mohapatra DP, and Sethi S (2014). A survey of computational intelligence approaches for software reliability prediction. ACM SIGSOFT Software Engineering Notes, 39(2): 1-10.

Boehm BW (1984). Software engineering economics. IEEE Transactions on Software Engineering, 10(1): 4-21.

Briand LC and Wieczorek I (2002). Resource estimation in software engineering. In: Marciniak JJ (Ed.), Encyclopedia of software engineering: 1160-1196. John Wiley and Sons, Hoboken, New Jersey, USA.

Briand LC, El Emam K, Surmann D, Wieczorek I, and Maxwell KD (1999). An assessment and comparison of common software cost estimation modeling techniques. In the 21st international conference on Software engineering, ACM, Los Angeles, USA, 313-322.

Clark B, Devnani-Chulani S, and Boehm B (1998). Calibrating the COCOMO II post-architecture model. In the 20th international conference on Software engineering, IEEE Computer Society, Kyoto, Japan, 477-480.

Costagliola G, Di Martino S, Ferrucci F, Gravino C, Tortora G, and Vitiello G (2006). Effort estimation modeling techniques: a case study for web applications. In the 6th international conference on Web engineering, ACM, New York, USA: 9-16.

Finnie GR, Wittig GE, and Desharnais JM (1997). A comparison of software effort estimation techniques: using function points with neural networks, case-based reasoning and regression models. Journal of Systems and Software, 39(3): 281-289.

Halstead MH (1977). Elements of software science: Operating and programming systems series. Elsevier Science Inc., New York, USA.

Hill P (2010). Practical software project estimation. Tata McGraw-Hill Education, New York, USA.

Martin CL, Pasquier JL, Yanez CM, and Tornes AG (2005). Software development effort estimation using fuzzy logic: a case study. In the 6th Mexican International Conference on Computer Science, IEEE, Puebla, Mexico: 113-120.

Mendes E (2007). Predicting web development effort using a bayesian network. In the 11th International Conference on Evaluation and Assessment in Software Engineering, British Computer Society: 83-93.

Mukhopadhyay T, Vicinanza SS, and Prietula MJ (1992). Examining the feasibility of a case-based reasoning model for software effort estimation. MIS Quarterly, 16(2): 155-171.

Panda A (2015). Effort estimation of agile and web-based software using artificial neural networks. M.Sc. Thesis, Department of Computer Science and Engineering National Institute of Technology, Rourkela, India.

Putnam LH (1978). A general empirical solution to the macro software sizing and estimating problem. IEEE Transactions on Software Engineering, SE4(4): 345-361.

Qamar N, Akhtar N, and Younas I (2018). Comparative analysis of evolutionary algorithms for multi-objective travelling salesman problem. International Journal of Advanced Computer Science and Applications, 9(2): 371-379.

Reddy S, Raju K, Srinivas T, and Devi GL (2007). A neural network approach for web cost estimation. In the 11th IASTED International Conference on Software Engineering and Applications, ACTA Press, Calgary, Canada: 37-41.

Reifer DJ (2000). Web development: estimating quick-to-market software. IEEE Software, 17(6): 57-64.

Richard L (1987). An introduction to computing with neural nets. IEEE Assp Magazine, 4(2): 4-22.

Ruhe M, Jeffery R, and Wieczorek I (2003). Cost estimation for web applications. In the 25th International Conference on

Software Engineering, IEEE Computer Society, Kyoto, Japan: 285-294.

Saleh K (2017). Global online retail spending – statistics and trends. Available online at: https://www.invespcro.com/blog/global-online-retail-spending-statistics-and-trends

Santani D, Bundele M, and Rijwani P (2014). Artificial neural networks for software effort estimation: A review. International Journal of Advances in Engineering Science and Technology, 3(3): 193-200.

Sentas P, Angelis L, Stamelos I, and Bleris G (2005). Software productivity and effort prediction with ordinal regression. Information and Software Technology, 47(1): 17-29.

Sharma R (2013). Survey: On algorithmic models for estimating software effort. European International Journal of Science and Technology, 2(3): 164-169.

Sheta A, Rine D, and Ayesh A (2008). Development of software effort and schedule estimation models using soft computing techniques. In the IEEE Congress on Evolutionary Computation: IEEE World Congress on Computational Intelligence, IEEE, Hong Kong, China: 1283-1289.

Singh BK and Misra AK (2012). Software effort estimation by genetic algorithm tuned parameters of modified constructive cost model for nasa software projects. International Journal of Computer Applications, 59(9): 22-26.

Srinivasan K and Fisher D (1995). Machine learning approaches to estimating software development effort. IEEE Transactions on Software Engineering, 21(2): 126-137.

Tronto BIF, da Silva JDS, and Sant'Anna N (2008). An investigation of artificial neural networks based prediction systems in software project management. Journal of Systems and Software, 81(3): 356-367.

Zhong S, Khoshgoftaar TM, and Seliya N (2004). Analyzing software measurement data with clustering techniques. IEEE Intelligent Systems, 19(2): 20-27.