Performance Enhancement of XML Parsing by using Artificial Neural Network

Yugandhara V. Dhepe
Dept. of Computer Science and Engineering
Prof. Ram Meghe Institute of Technology and
Research Badnera, Amravati University, Maharashtra, India

Dr. G.R. Bamnote
Dept. of Computer Science and Engineering
Prof. Ram Meghe Institute of Technology and
Research Badnera, Amravati University, Maharashtra, India

Abstract - XML is used for data representation and exchange. XML data processing becomes more and more important for server workloads like web servers and database servers. One of the most time consuming part is XML document parsing. Parsing is a core operation performed before an XML document can be navigated, queried, or manipulated. Recently, high performance XML parsing has become a topic of considerable interest. This paper proposes a mechanism for efficiently processing XML documents with the help of Artificial Neural Network (ANN). Provide the set of XML documents to the system and parsing results will store in the database. With the help of Artificial Neural Network (ANN) the performance of XML parsing will improve by reducing parsing time of XML document. Levenberg–Marquardt algorithm (LMA) is used to train the neurons in the Artificial Neural Network (ANN) to recognize XML document pattern. This proposed system will improve the performance of xml parsing by reducing parsing time of XML document with the help of Artificial Neural Network (ANN).

Keywords - XML (extensible markup language) , SAX parser, Artificial Neural Network (ANN).

I. Introduction

XML stands for Extensible Markup Language (XML). It is used to store and exchange structured information. It was designed to provide flexible information identification in web documents [1]. XML data processing becomes an important workload for web servers, database servers, etc.
Studies have shown that these servers spend a significant portion of their execution time in XML data processing [2], especially in XML data parsing. Data parsing is to convert the input XML document and break it into small elements. It is one of the most important portions in XML data processing because an XML document has to be parsed before any other operations can be performed [3].
There are mainly two categories of XML programming interfaces, DOM (Document Object Model) and SAX (Simple API for XML). DOM is a tree-based interface and SAX is an event-driven interface [4]. SAX has advantages over DOM it gain tremendous speed and use very little memory.
The Artificial Neural Network (ANN) technology is built up with an inspiration of functioning of human nervous system. Artificial neural network is an information processing devices, which are built from interconnected elementary processing elements called neurons. Learning involves adjustments to the synaptic connection known as weights that exist between the neurons [5]. The Levenberg–Marquardt algorithm (LMA) is used to train the neurons [6].
The paper is organised as, in section 1 above introduces XML parsing and Artificial Neural Network (ANN) technology. In section 2 Literature review for XML, XML parsing, Artificial Neural Network (ANN) and Levenberg-Marquardt (LM) algorithm is given. In section 3 proposed works is discussed. Finally concluded in section 4.

II. Literature Survey

Parsing is a core operation performed before an XML document can be navigated, queried, or manipulated. Recently, high performance XML parsing has become a topic of considerable interest.

Bruno Oliveira, Vasco Santos and Orlando Belo analyses that due to the simplicity of its hierarchical structure, XML (Extensible Markup Language) are widely used for data representation in many applications. As a result of its portability, XML is used to ensure data interchanging among systems with high heterogeneous natures, facilitating data communication and sharing, its platform independent, which makes it quite attractive for the majority of applications [7].

Tong, T. et al and Kiselyov, O. studied the XML parser. They analyses the work of XML parser. XML parser can read the XML document components via Application Programming Interfaces (APIs) in two approaches. For stream-based approach (also known as event-based parser), it reads through the document and signal the application every time a new component appears. As for tree-based approach, it reads the entire document into a memory resident collection of object as a representation of original document in tree structure [8, 9]. As a result, tree-based approach is not suitable for large-scale XML data because it can easily run out of memory.

Elliotte Rusty Harold studied the various API for XML parsing. Starting in Java 1.4, Sun bundled the Crimson XML parser and the SAX2, DOM2, and TrAX APIs into the standard Java class library. They also threw in a couple of factory classes, and called the whole thing the “Java API for XML Processing” (JAXP) [11]. Putting them all together, JAXP can replace most of the parser-dependent part of DOM [12].

Pranob K Charles, Dr. H.Khan ,Ch.Rajesh Kumar ,N.Nikhita Santhosh Roy ,V.Harish ,M.Swathi analyses Artificial Neural Network. Artificial Neural Network is the simple clustering of the primitive artificial neurons. This clustering occurs by creating layers, which are then connected to one another. There are usually a number of hidden layers between these two layers. When the input layer receives the input its neurons produce output, which becomes input to the other layers of the system. The process continues until a certain condition is satisfied or until layer is invoked and fires their output to the external environment [5].

A.Saravanan, Dr.P.Nagarajan analyses the various types of training algorithms for which the Multilayer Perception (MLP) Network learn exist. This depends on many factors including the number of weights and biases, error goal and number of training iterations (epochs). Back propagation algorithm, the common and most widely used algorithm in training Artificial Neural Network learns by calculating an error between desired and actual output and propagate the error information back to each node in the network. In this study, two training algorithms are evaluated for the different dataset allocations into training, validation and testing. They are: Levenberg-Marquardt and Quasi-Newton [6].

Amir Abolfazl Suratgar, Mohammad Bagher Tavakoli, and Abbas Hoseinabadi analyses the Error Back Propagation (EBP) algorithm. The Error Back Propagation (EBP) algorithm has been a signification improvement in neural network research, but it has a weak convergence rate. Many efforts have been made to speed up EBP algorithm. All of these methods lead to little acceptable results. The Levenberg-Marquardt (LM) algorithm ensued from development of EBP algorithm dependent methods [13].

III. Proposed Work

A. Parsing Process

Parsing is the process of reading a document and dissecting it into its elements and attributes, which can then be analyzed. In XML, parsing is done by an XML processor, the most fundamental building block of a Web application [14]. All modern browsers have a built-in XML parser. An XML parser converts an XML document into an XML SAX object, which can then be manipulated with a JavaScript [10]. As the document is parsed, the data in the document becomes available to the application using the parser. The XML processor parses and generates an XML document. The application uses an API to access objects that represent part of the XML document [15].

B. SAX Parser

As each event occurs, the program calls the appropriate event handler. The event handlers work like the functions of a graphical interface, which is also event-driven in that one function handles a mouse click in one button, another handles a key press, and so on. In the case of SAX, each event handler processes an event such as the beginning of an element or the appearance of a processing instruction. The Parser Factory object creates a framework around the parser of your choice. It parses the document, calling on the Document Handler, Entity Resolver, DTD Handler, and Error Handler interfaces as necessary [16].

In Java, an interface is a collection of routines, or methods in a class. The document-handler interface is where you put the code for your program. Within the document handler, you must implement methods to handle elements, attributes, and all the other events that come from parsing an XML document [7].

C. Training Algorithms

Various types of training algorithms for which the multilayer perception (MLP) network learn exist but it is very difficult to know which training algorithm will be suitable for training ANN model. This depends on many factors including the number of weights and biases, error goal and number of training iterations (epochs). There are two training algorithms for the different dataset allocations into training, validation and testing. They are: Levenberg-Marquardt and Quasi-Newton.

D. Levenberg-Marquardt Artificial Neural Networks

In an ANN, each neuron will typically apply an activation function to a weighted sum of inputs and provide a single output. Such a training process can be performed by one of a number of algorithms, the most popular being back propagation [17], but LM and Conjugate Gradient Descent [18] are also in common use. An example MLP network with 11 input neurons, four hidden neurons and two output neurons is shown in Figure 1. In the general case, for *M* input neurons *im*, *P* hidden neurons *hp* and one output neuron *o*, the weights on the edges between the input and hidden layers can be represented by *Wpm* and those between the hidden and output layer (assuming a single output neuron) by *wp*. Given *k* input vectors, input value *m* is given the value when presented with vector *γ* where *γ*= {1,2,…,*k*}. m i .

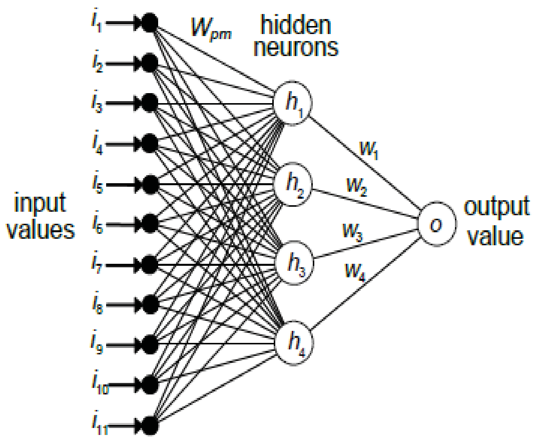

Figure 1: Example of an MLP ANN with a single output

The LM algorithm has recently become increasingly popular as its second-order optimization techniques allow a very efficient batch update method. A drawback of the LM approach is that the ANN must have only a single output, but this can be overcome by implementing multiple networks [19]. For the detailed mathematics underlying the LM algorithm, Marquardt [20], but general LM algorithm is shown in Figure 2 and is briefly explained below.

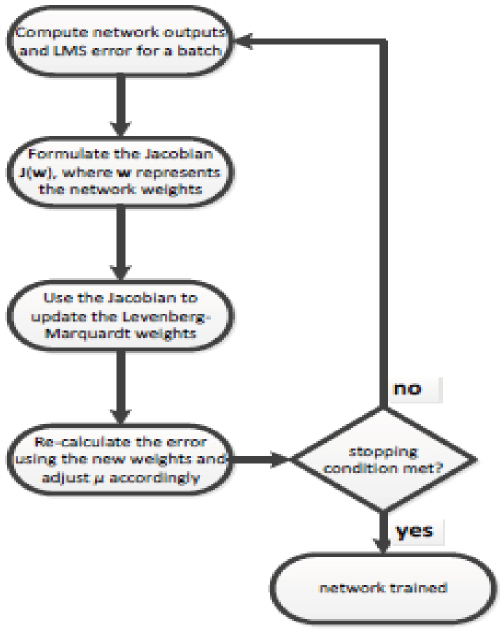
Compute network outputs and LMS error for a batch
Formulate the Jacobian J(w), where w represents the network weights
Use the Jacobian to update the Levenberg-Marquardt weights
Re-calculate the error using the new weights and adjust μ accordingly
stopping condition met?
no
yes
network trained

Figure 2: Flow diagram outlining the procedure for using the Levenberg- Marquardt training algorithm.

Each batch of data is fed forward through the network, as described previously, to obtain a vector of output values each calculated using equation (1) below, where *z* is the activation function.

$$o^{\gamma} = z\left(\sum_{p=1}^{P} z\left(\sum_{m=1}^{M} i_m^{\gamma} W_{pm}\right) w_p\right) \qquad (1)$$

The least mean-square (LMS) error can then be obtained using

$$E[w] = \frac{1}{2}\sum_{\gamma=1}^{k}(R^{\gamma} - o^{\gamma})^2 \ , \qquad (2)$$

Where *Rγ* is the desired output from the ANN for a specific input vector γ. The Jacobian matrix used in LM requires a vector of all the weights contained within the network to calculate a matrix of partial derivatives (with respect to each weight individually) for each input pattern in the batch. The Jacobian is given by

$$\mathbf{J}(\mathbf{w}) = \begin{pmatrix} \frac{\partial e_1(\mathbf{w})}{\partial w_1} & \cdots & \cdots & \frac{\partial e_1(\mathbf{w})}{\partial w_v} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_\gamma(\mathbf{w})}{\partial w_1} & \cdots & \cdots & \frac{\partial e_\gamma(\mathbf{w})}{\partial w_v} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_p(\mathbf{w})}{\partial w_1} & \cdots & \cdots & \frac{\partial e_p(\mathbf{w})}{\partial w_v} \end{pmatrix}, \quad (3)$$

Where $v = MP + 2P$ and **w** is a vector of weights **w** = [W11 ,.., *WPM*, *B*1, .. ,*BP*, *w*1,..,*wP*]T, where the *Bp* values are the bias values of the hidden neurons. To update the weights during training the LM algorithm determines a weight update vector **Δw**, calculated by

$$\Delta\mathbf{w} = \left[\mathbf{J}^{\mathrm{T}}(\mathbf{w})\mathbf{J}(\mathbf{w}) + \mu\mathbf{I}\right]^{-1}\mathbf{J}^{\mathrm{T}}(\mathbf{w})\mathbf{e}\,, \quad (4)$$

Where **e** is the vector containing errors for each input vector in the batch and **I** is the identity matrix of dimension *v*. The new weights can now be calculated by

$$\mathbf{w}_{\mathbf{new}} = \mathbf{w}_{\mathbf{old}} + \Delta\mathbf{w}\ . \quad (5)$$

The ANN's LMS error with new weights is now computed. If the new error is smaller than the previous error then μ is reduced by a factor μ-, but if the error is larger than the previous error u is increased by a factor of μ+. The values of μ, μ- and μ+ are all training parameters that must be selected before training the network [21].

IV.CONCLUSION

The mechanism is used in this paper for efficiently processing XML documents. Provide set of XML document to SAX parser as an input, parsing results is stored in the database and calculate the time required to parse the set of XML document. Providing the same set of XML document to the SAX parser which is used Artificial Neural Network (ANN) and also calculate the time required to parse the set of XML document. If time required for parsing the set of XML document to SAX parser and SAX which is used Artificial Neural Network (ANN) is compared. Then SAX parser which is used Artificial Neural Network (ANN) required less time for parsing the set of XML document than simple SAX parser. This proposed system improves the performance of XML parsing by reducing parsing time of set of XML document with the help of Artificial Neural Network (ANN).

REFERENCES

[1] W3C, "EXTENSIBLE MARKUP LANGUAGE (XML)." [ONLINE]. AVAILABLE: HTTP://WWW.W3.ORG/XML.

[2] W3C, "Document object model (DOM) level 2 core specification." [Online]. Available: http://www.w3.org/TR/DOM-Level-2-Core.

[3] Fangju Wang, Jing Li, Hooman Homayounfar, "A space efficient XML DOM parser" in Journal of Data & Knowledge Engineering,Volume 60, Issue 1, January 2007, Pages 185-207.

[4] Chengkai Li, "XML Parsing, SAX/DOM". Department of Computer Science and Engineering, University of Texas at Arlington.

[5] Pranob K Charles, Dr. H.Khan, Ch.Rajesh Kumar, N.Nikhita Santhosh Roy, V.Harish, M.Swathi " Artificial Neural Network based Image Compression using Levenberg-Marquardt Algorithm" International Journal of Modern Engineering Research (IJMER) www.ijmer.com Vol.1, Issue.2, pp-482-489 ISSN: 2249-6645.

[6] A.Saravanan, Dr.P.Nagarajan, "Performance of ANN in Pattern Recognition For Process Improvement Using Levenberg- Marquardt And Quasi-Newton Algorithms" IOSR Journal of Engineering (IOSRJEN) e-ISSN: 2250-3021, p-ISSN: 2278-8719 Vol. 3, Issue 3 (Mar. 2013).

[7] Bruno Oliveira, Vasco Santos and Orlando Belo, "Processing XML with Java – A Performance Benchmark". International Journal of New Computer Architectures and their Applications (IJNCAA) 3(1): 72-85. The Society of Digital Information and Wireless Communications (SDIWC) 2013 (ISSN: 2220-9085).

[8] Tong, T. et al, “Rules about XML in XML”, Expert Systems with Applications, Vol. 30, No.2, 2006, pp. 397-411.

[9] Kiselyov, O., “A better XML parser through functional programming”, LNCS 2257, 2002, pp. 209-224.

[10] F. Wang, J. Li, and H. Homayounfar, “A space efficient XML DOM parser”, Data & Knowledge Engineering, vol. 60, no. 1, pp. 185–207, 2007.

[11] E. Perkins, M. Kostoulas, A. Heifets, M. Matsa, and N. Mendelsohn, “Performance Analysis of XML APIs”, in XML 2005 Conference proceeding, 2005.

[12] Elliotte Rusty Harold , “Processing XML with JAVA – A Guide to SAX, DOM, JDOM, JAXP and TrAX.” 2 0 0 3 - 0 1 - 2 4.

[13] Amir Abolfazl Suratgar, Mohammad Bagher Tavakoli, and Abbas Hoseinabadi, “Modified Levenberg-Marquardt Method for Neural Networks Training” World Academy of Science, Engineering and Technology 6 2005.

[14] Wei Zhang van Engelen, R.A., “High-Performance XML Parsing and Validation with Permutation Phrase Grammar Parsers”, ICWS '08. IEEE International Conference on Web Services, 2008, Beijing, pp 286 – 294.

[15] Girish Tere and Bharat Jadhav “Efficient Processing of XML Documents” International Conference on Technology Systems and Management (ICTSM) 2011 Proceedings published by International Journal of Computer Applications (IJCA).

[16] T. C. Lam, J. J. Ding and J.-C. Liu, “XML Document Parsing: Operational and Performance Characteristics”, Computing & Processing, vol. 41, no. 9, pp. 30–37, 2008.

[17] J. Hertz, A. Krogh, R.G.Palmer, “Introduction to the theory of neural computation.” (Reading, MA: Addison-Wesley, 1991), 115-120. [21] M. T. Hagan. and M. Menhaj, “Training feed-forward networks with the Marquardt algorithm.” IEEE Trans. Neural Networks, 5(6), 1994, 989-993.

[18] C. Charalambous, “Conjugate gradient algorithm for efficient training of Artificial Neural Networks.” IEE Proc. G Circuits, Devices and Systems, 139(3), 1992, 301-310.

[19] D. J. Mulvaney and I. P. W. Sillitoe, “The classification of ultrasonic signals using novel neural network approaches.” Int. J. Robotics and Automation, 14(1), 1999. 15-22.

[20] D. W. Marquardt, “An algorithm for least-squares estimation of nonlinear parameters.” J. Soc. Industrial and Applied Mathematics, 11(2), 1963, 431-441.

[21] David Scanlan, 1 David Mulvaney, “Graphics Processor Unit Hardware Acceleration of Levenberg-Marquardt Artificial Neural Network Training.” Research Inventy: International Journal Of Engineering And Science Issn: 2278-4721, Vol. 2.