Research Journal of Applied Sciences, Engineering and Technology 7(17): 3643-3648, 2014
DOI:10.19026/rjaset.7.717
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: November 22, 2014 Accepted: December 04, 2013 Published: May 05, 2014

Research Article

Optimizing Support Vector Machine for Classifying Non Functional Requirements

[1]K. Mahalakshmi, [2]R. Prabhakar and [3]V. Balakrishnan
[1]Department of Computer Science Engineering, School of Engineering and Technology, Surya Group of Insitutions, Villupuram, India
[2]Department of Computer Science Engineering, Coimbatore Institute of Technology, Coimbatore, India
[3]Department of Management Studies, DDE, Annamalai University, Chidabaram, India

Abstract: Problems faced in contemporary practice should be understood to improve requirements engineering processes. System requirements are descriptions of services provided by a system and operational constraints. Non-Functional Requirements (NFR) defines overall qualities/attributes of the system. NFR analysis is a significant activity in this branch of engineering. In this study, a methodology for classifying NFR is presented. Inverse Document Frequency is used for extracting the features from the NFR dataset and is classified by Support Vector Machine (SVM). The efficiency of the SVM depends upon the parameter used with Radial Basis Function. In this study, the RBF kernel is optimized by Artificial Bee Colony algorithm (ABC) to optimize the RBF parameters to improve performance.

Keywords: Artificial Bee Colony algorithm (ABC), functional requirements, Non-Functional Requirements (NFR), requirement engineering, Support Vector Machine (SVM)

INTRODUCTION

Requirements Engineering (RE) is a software engineering branch dealing with real-world goals for, functions of and software systems (Aljahdali *et al*., 2011) constraints. This process is arguably a most important process in software development. Much knowledge is built up in RE's early phase, supporting reasoning about organizational alternatives, objectives, implications etc.

Software requirements are requirements related to categories in business, functional, non-functional, performance and security (Selvakumar and Rajaram, 2011). Functional requirements reveal how products work. Functional requirements confusion affects product's functionality requiring that inconsistency be removed at the beginning. As functional requirements deal with specific issues, they are implemented through specific localized modules/components (Cysneiros and Yu, 2004). Though stated informally, they can be formalized when needed.

Non Functional Requirements (NFRs) are known as Quality Requirements and in contrast to Functional Requirements, NFRs reveal system constraints and specific ideas system qualities like usability, accuracy, performance, safety, reliability and security (Nuseibeh and Easterbrook, 2000). This results in NFRs being linked to Functional Requirements. Non-functional requirements are harder to express measurably, making their analysis harder. NFRs are system properties as a whole and so cannot be substantiated for individual components verification.

Similar to functional requirements, system success is dependent on adherence to NFRs. Costly issues arise (Slankas and Williams, 2013) when NFRs are ignored or missed. Due to the need to analyze and implement NFRs from various resources, system analysts should identify and categorize NFRs quickly. As NFRs existed from the inception of software engineering, there is no consensus for a name or the NFR identification. They are just referred to as the "ilities," a system's quality aspects. Others label NFRs as systemic requirements.

NFR elicitation and analysis involve various people in organizations. The word 'stakeholder' refers to any person/group affected directly or indirectly by the system. Single view ensures a look at requirements from a specific perspective. To elicit and analyze requirements multiple views must be considered to meet stakeholder expectations. Non-functional requirements for general four layered analysis architecture are seen in Fig. 1 (Rao and Gopichand, 2012).

NFRs are usually seen relatively late in development processes (Cleland-Huang *et al*., 2006). Stakeholder's quality constraints from requirements gathering process is documented through many artifacts like memos, interview notes and meeting minutes with analysts generally failing to get a clear picture on system-wide NFR. NFR-Classifiers detect and classify

Corresponding Author: K. Mahalakshmi, Department of Computer Science Engineering, School of Engineering and Technology, Surya Group of Insitutions, Villupuram, India

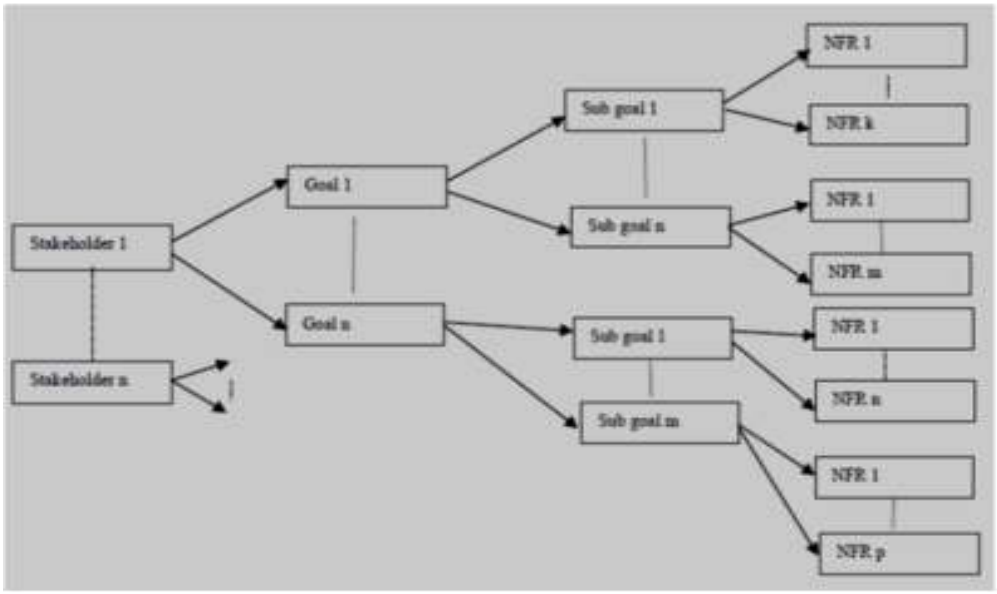

Fig. 1: General architecture for four layered analysis for non-functional requirements

early aspects which are a concern for cross-cutting dominant system decomposition found in requirements specification or early design documents. Many "early aspects" are similar to high-level NFRs like performance, security, portability and usability. Intermediate level aspects can be logging and authentication and lower-level concerns focus on programmatic concepts like buffering and caching. The NFR classification helps the developers to implement changes and to cross check the quality of the system.

In the current work, Support Vector Machine (SVM) with different kernels such as polykernel and Radial Basis Function (RBF) are used to classify the NFR. The performance of the RBF kernel is dependent upon the parameters C and gamma (γ). The optimum values of these parameters are formulated as optimization problem. "Swarm intelligence" and genetic algorithm are widely used for optimization, to maximize or minimize the cost function. The most common swarm optimizations are based on the collective behavior of the ants, fish schooling or social behavior of bird flocking. In this study, Artificial Bee Colony algorithm (ABC) is implemented for parameter selection of the RBF kernel of SVM.

The ABC algorithm is based on foraging behaviour of honeybee swarm (Karaboga and Basturk, 2008). Unlike genetic algorithm, ABC does not necessitate crossover rate and mutation rate to obtain solution. The performance of the ABC algorithm is better when compared with differential evaluation, PSO, ACO and an evolutionary algorithm (Karaboga and Basturk, 2008). ABC's are simple, flexible to implement and also have the advantage of requiring less control parameters when compared to other optimization algorithms (Bolaji *et al*., 2013).

In this study, a methodology for classifying NFR is presented. NFR dataset available in the promise data repository is used to evaluate the proposed methodology. Features are extracted using Inverse Document Frequency (IDF) from the NFR dataset and is classified by Support Vector Machine (SVM). The efficiency of the SVM depends upon the parameter C and gamma (γ) used with Radial Basis Function. In this study, the RBF kernel is optimized by Artificial Bee Colony algorithm (ABC) to optimize the RBF parameters to improve performance.

LITERATURE REVIEW

A framework for conflict analysis among NFRs using integrated analysis of functional and non-functional requirements was provided by Sadana and Liu (2007). This identifies conflicts and analyzes them based on relationships between quality attributes, constraints and functionalities. The proposed framework's output was a conflict hierarchy refining conflicts among non-functional requirements level-wise. A case study was finally provided where the framework analyzed and detected conflicts among a search engine's NFRs.

NFR and its requirements and importance in various fields was surveyed by Bajpai and Gorthi (2012) who also investigated the effect of working for NFRs leading to discovery of new Functional Requirements. Efforts were made to show recent research in finding, specifying and rectifying Non-Functional attributes.

The first effort to seek key NFR challenges and issues at RE elicitation level was taken by Ullah *et al*. (2011). A primary RE engineer's responsibility is elicitation of non-functional and functional requirements. Functional requirements reveal functionality to be performed while NFRs compel restrictions on such functionality. The Software

industry has raised the demand for not only required functionality but NFRs necessity like security, performance, security, usability and privacy. The authors discovered approaches and methods suggested in literature to offset such issues.

Taxonomy for NFR in a service-oriented context was presented by Galster and Bucherer (2008) which implemented three NFR categories: process requirements, non-functional external requirements and non-functional service requirements. The taxonomy was capable of being applied to individual services and service-based systems. The taxonomy was a starting point and checklist to handle non-functional issues in service-oriented, specifically distributed environments.

A model-based approach to focus on NFR in exploring its effect on system design process was proposed by Tsadimas *et al*. (2009). A requirement model representing how NFRs were related between them and to system components in overall system architecture was defined. SysML was adopted as modeling language, as it enabled requirement definition and could be extended. Also, requirement derivation process was discussed and case studies where new concepts were applied practically, while redesigning a large-scale organization's legacy system was presented.

Gokyer *et al*. (2008) introduced the use of a Machine Learning (ML) and natural language processing techniques methods to relate NFRs to classified "architectural concerns". The machine learning used in this study was SVM. A charted roadmap and the automated requirements engineering toolset for this roadmap were demonstrated. The approach was validated and the effectiveness of the toolset on the snapshot was demonstrated with a real-life project.

METHODOLOGY

Inverse Document Frequency (IDF) measures a word's importance. IDF appears in many heuristic measures in information retrieval but, IDF has itself not been a heuristic till now. It is defined as the logarithm of the ratio of documents number containing a given word meaning that rare words have high IDF with common function words like "the" having low IDF (Robertson, 2004). IDF measures a word's ability to differentiate between documents. Text Classification assigns text documents to pre-defined classes set through a machine learning technique automatically. Classification is usually based on the text document's significant words or key-features. It is a supervised machine learning task as classes are pre-defined.

IDF represents scaling. When a term 'a' occurs in many documents frequently, its importance is scaled down due to lowered discriminative power. The IDF (a) is defined as follows:

$$IDF(a) = \log \frac{1+|x|}{x_a} \quad (1)$$

Term frequency and inverse document frequency are the two scores of TF-IDF. Term frequency is counting the times a term occurs in a specific document, while IDF is achieved through dividing total documents by documents where a specific word is repeated. Multiplying these values results in a high score for repeatedly and frequently occurring words in few documents. Scores are low for terms appearing frequently in many documents.

Support Vector Machine (SVM): SVMs suit classification tasks related to and containing elements of non-parametric applied statistics, neural networks and machine like classical techniques. SVMs through nonlinear mapping transform original training data into a higher dimension (Vishwanathan and Narasimha Murty, 2002). Data from two classes separated by a hyperplane is mapped nonlinearly. SVM locates the hyperplane using support vectors and margins. The margin is the distance between hyperplane and entity. SVM's advantage is that it is highly accurate and not liable to over-fitting. Its disadvantage is that it is time consuming. SVM with input vector and normal vector to hyperplane, leads to output u given by:

$$u = \vec{w}.\vec{x} - b \quad (2)$$

A kernel function is defined as $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$. The Radial Basis function is given as follows:

$$K(x_i, x_j) = \exp(-\gamma \| x_i - x_j \|^2), \gamma > 0 \quad (3)$$

There are two parameters to be determined in RBF kernel are C and gamma (γ) and proper parameter setting of these parameters improves SVM classification accuracy. The C and γ impact the learning performance of the SVM (Kuba *et al*., 2002). The γ parameter defines the distance a single training example can reach and the C parameter trades off training examples misclassification against decision surface simplicity. Experiments are undertaken to evaluate SVM performance through variations of the γ and C parameters and by optimizing it by ABC.

Artificial Bee Colony (ABC) algorithm is a recently introduced swarm-based algorithm in which the position of food source represents a possible solution to an optimization issue and a food source's nectar amount corresponds to the associated solution's quality (fitness) (Karaboga and Akay, 2009). The number of employed bees or onlooker bees equals number of solutions in a population.

To begin with, ABC generates randomly distributed initial population P (C = 0) of SN solutions (food source positions), with SN denoting employed or onlooker bees size. Each solution xi (1, 2… SN) is a D-dimensional vector where D is optimization parameters number. After initialization, the positions (solutions) population is subject to repeated cycles, C = 1, 2… MCN, of search processes of employed, onlooker and scout bees.

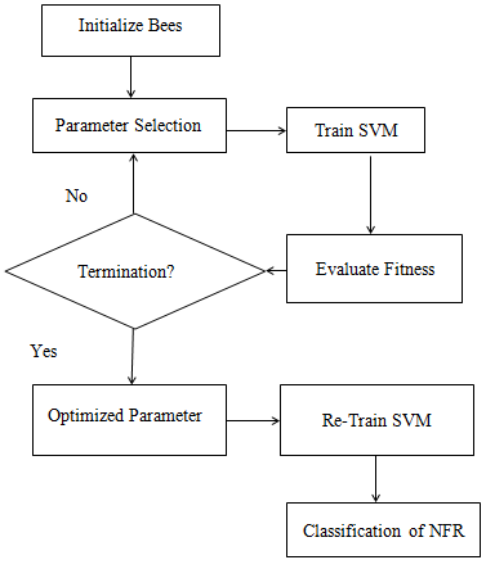

Fig. 2: Flowchart-ABC optimizing SVM

The pseudo-code of the ABC algorithm (Kumbhar and Krishnan, 2011):

1: Initialize the population of solutions xi, i = 1... SN
2: Evaluate the population xi, Gi, i = 1, ..., NP
3: For cycle = 1 to MCN do
4: Produce new solutions vi for the employed bees and evaluate them.
5: Apply the greedy selection process.
6: Calculate the probability values pi for the solutions xi
7: Produce the new solutions vi for the onlookers from the solutions xi selected depending on pi and evaluate them.
8: Apply the greedy selection process.
9: Determine the abandoned solution for the scout, if exist and replace it with a new randomly produced solution xi
10: Memorize the best solution achieved so far.
11: Cycle = cycle + 1
12: End for

To optimize the parameters C and γ, ABC is adapted to execute the search for optimal combination of (C, γ). The objective function is based on Root Mean Squared Error (RMSE) attained by the SVM-RBF. Thus, the ABC discovers the combination of (C, γ) with the lowest RMSE to optimize the performance. Figure 2 shows the steps involved in the proposed methodology.

RESULTS AND DISCUSSION

NFR dataset available in the promise data repository is used to evaluate the proposed methodology. The NFR dataset consists of 15 requirement specifications of MS student projects with a total of 326 NFRs and 358 FRs. NFR categories included availability, scalability, usability and security. Features are extracted using IDF and is classified by SVM with various kernels and ABC optimized SVM. The experiments are conducted with: SVM with Polykernel, SVM with RBF Kernel gamma = 0.1, SVM with RBF Kernel gamma = 0.5, SVM with RBF Kernel gamma = 1 and SVM with ABC optimization. It is observed that SVM with ABC optimization does better than all the other classifiers. Table 1 tabulates the classification accuracy and RMSE for all the techniques used. Figure 3 depicts the classification accuracy for all the techniques used.

Table 1: Classification accuracy

Classifier	Classification accuracy	RMSE
SVM with polykernel	81.25	0.2864
SVM with RBF kernel gamma = 0.1	82.211	0.2714
SVM with RBF kernel gamma = 0.5	83.01	0.2687
SVM with RBF kernel gamma = 1	82.53	0.2706
SVM with ABC optimization	85.1	0.2312

Table 2: Precision, recall and F-measure

Classifier	Precision	Recall	F-measure
SVM with polykernel	0.897	0.897	0.897
SVM with RBF kernel gamma = 0.1	0.898	0.914	0.906
SVM with RBF kernel gamma = 0.5	0.900	0.931	0.915
SVM with RBF kernel gamma = 1	0.898	0.914	0.906
SVM with ABC optimization	0.917	0.948	0.932

Classification Accuracy SVM with ABC optimization does better by 4.52% than SVM with polykernel, by 3.39% than SVM with RBF Kernel gamma = 0.1, by 2.45% than SVM with RBF Kernel gamma = 0.5 and by 3.01% than SVM with RBF Kernel gamma = 1. Figure 4 depicts the RMSE for all the techniques used.

SVM with proposed ABC optimization decreases the RMSE by 19.27% than SVM with polykernel and by 14.81% than SVM with RBF Kernel gamma = 0.1, by 13.95% than SVM with RBF Kernel gamma = 0.5 and by 14.56% than SVM with RBF Kernel gamma = 1. Table 2 tabulates the Precision and Recall for all the techniques used. Figure 5 depicts the Precision and Recall for all the techniques used.

It is observed from Fig. 5 that the Precision achieved by SVM with ABC optimization is better by 2.18% than SVM with polykernel, by 2.07% than SVM with RBF Kernel gamma = 0.1, by 1.85% than SVM with RBF Kernel gamma = 0.5 and by 2.07% than SVM with RBF Kernel gamma = 1. Similarly, for Recall SVM with ABC optimization does better by 5.37% than SVM with polykernel, by 3.58% than SVM with RBF Kernel gamma = 0.1, by 1.79% than SVM with RBF Kernel gamma = 0.5 and by 3.58% than SVM with RBF Kernel gamma = 1. Figure 6 depicts the F-Measure for all the techniques used.

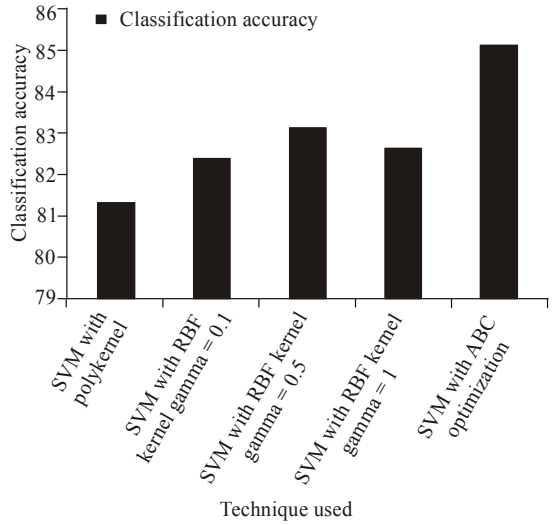

Fig. 3: Classification accuracy

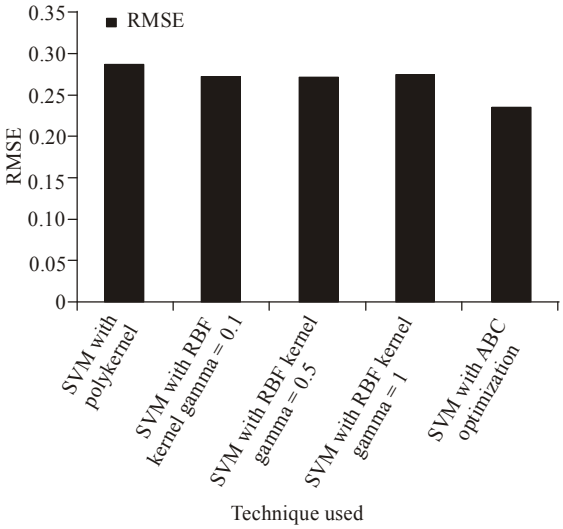

Fig. 4: RMSE

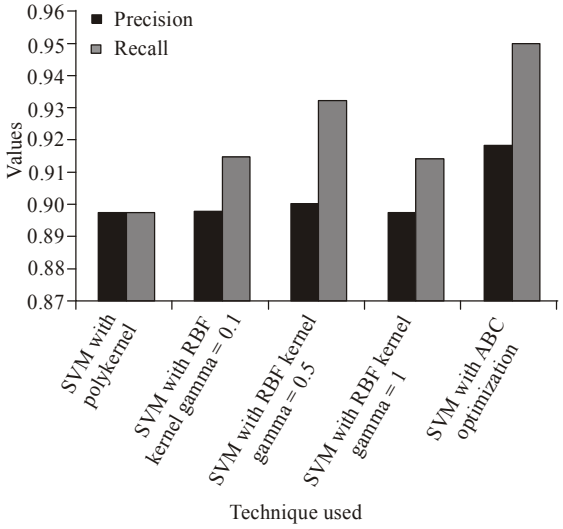

Fig. 5: Precision and recall

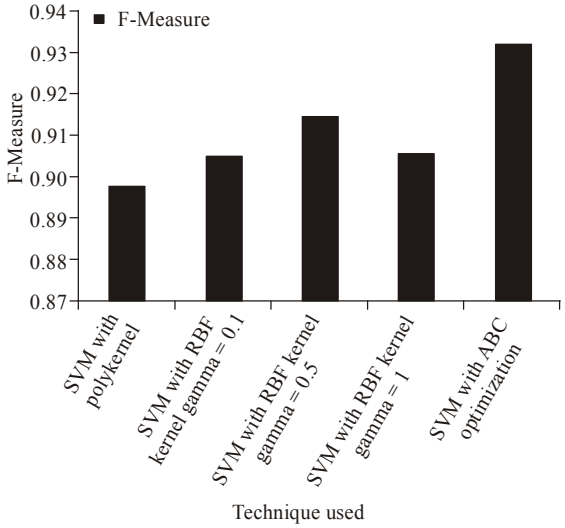

Fig. 6: F-measure

From Fig. 6 it is observed that the F-Measure achieved by SVM with ABC optimization is better by 3.75% than SVM with polykernel, by 2.78% than SVM with RBF Kernel gamma = 0.1, by 1.82% than SVM with RBF Kernel gamma = 0.5 and by 2.78% than SVM with RBF Kernel gamma = 1. So the proposed approach used in this research gives the better solution for the NFR analysis.

CONCLUSION

When RE is taken as a continual task in the project, RE process model is iterative and its activities ensued across multiple phases, making process models seem iterative. A preliminary step to determine NFRs is by identifying its type and categories. Operating conditions are the first (high-level) step in identifying business requirements. SVM is utilized as classifier in this approach with ABC being used for optimization. Experiments reveal the proposed approach's improved results and achieve better performance. Classification Accuracy achieved by SVM with ABC optimization is better by 2.45 to 4.52% than other SVM Kernels.

REFERENCES

Aljahdali, S., J. Bano and N. Hundewale, 2011. Goal oriented requirements engineering: A review. Proceeding of the International Conference on Computer Applications in Industry and Engineering, pp: 328-333.

Bajpai, V. and R.P. Gorthi, 2012. On non-functional requirements: A survey. Proceeding of the 2012 IEEE Students' Conference on Electrical, Electronics and Computer Science (SCEECS), pp: 1-4.

Bolaji, A.L., A.T. Khader, M.A. Al-Betar and M.A. Awadallah, 2013. Artificial bee colony algorithm, its variants and applications: A survey. J. Theor. Appl. Inform. Technol., 47(2): 434-439.

Cleland-Huang, J., R. Settimi, X. Zou and P. Solc, 2006. The detection and classification of non-functional requirements with application to early aspects. Proceeding of the IEEE 14th IEEE International Conference on Requirements Engineering, pp: 39-48.

Cysneiros, L.M. and E. Yu, 2004. Non-functional Requirements Elicitation. In: Perspectives on Software Requirements. Kluwer Academic Publishers, Boston, pp: 115-138.

Galster, M. and E. Bucherer, 2008. A taxonomy for identifying and specifying non-functional requirements in service-oriented development. Proceedings of the 2008 IEEE Congress on Services - Part I (SERVICES '08), pp: 345-352.

Gokyer, G., S. Cetin, C. Sener and M.T. Yondem, 2008. Non-functional requirements to architectural concerns: ML and NLP at crossroads. Proceeding of the 3rd International Conference on Software Engineering Advances (ICSEA '08), Oct. 26-31, pp: 400-406.

Karaboga, D. and B. Basturk, 2008. On the performance of Artificial Bee Colony (ABC) algorithm. Appl. Soft Comput., 8(1): 687-697.

Karaboga, D. and B. Akay, 2009. A comparative study of artificial bee colony algorithm. Appl. Math. Comput., 214(1): 108-132.

Kuba, P., P. Brazdil, C. Soares and A. Woznica, 2002. Exploiting sampling and meta-learning for parameter setting for support vector machines. Proceeding of the 8th Iberoamerican Conference on Artificial Intelligence, Learning and Data Mining Associated with Iberamia, pp: 209-216.

Kumbhar, P.Y. and S. Krishnan, 2011. Use of Artificial Bee Colony (ABC) algorithm in artificial neural network synthesis. Int. J. Adv. Eng. Sci. Technol., 11(1): 162-171.

Nuseibeh, B. and S. Easterbrook, 2000. Requirements engineering: A roadmap. Proceedings of the ACM Conference on the Future of Software Engineering, pp: 35-46.

Rao, A.A. and M. Gopichand, 2012. Four layered approach to non-functional requirements analysis. Int. J. Comput. Sci. Issues, 8(6): 371-379.

Robertson, S., 2004. Understanding inverse document frequency: on theoretical arguments for IDF. J. Doc., 60(5): 503-520.

Sadana, V. and X.F. Liu, 2007. Analysis of conflicts among non-functional requirements using integrated analysis of functional and non-functional requirements. Proceeding of the IEEE 31st Annual International Computer Software and Applications Conference (COMPSAC, 2007), 1: 215-218.

Selvakumar, J. and M. Rajaram, 2011. Performance evaluation of requirements engineering methodology for automated detection of non functional requirements. Int. J. Comput. Sci. Eng., 3(8).

Slankas, J. and L. Williams, 2013. Automated extraction of non-functional requirements in available documentation. Proceeding of the 1st International Workshop on Natural Language Analysis in Software Engineering (NaturaLiSE, 2013). San Francisco, CA.

Tsadimas, A., M. Nikolaidou and D. Anagnostopoulos, 2009. Handling non-functional requirements in information system architecture design. Proceeding of the IEEE 4th International Conference on Software Engineering Advances (ICSEA'09), pp: 59-64.

Ullah, S., M. Iqbal and A.M. Khan, 2011. A survey on issues in non-functional requirements elicitation. Proceeding of the IEEE 2011 International Conference on Computer Networks and Information Technology (ICCNIT), pp: 333-340.

Vishwanathan, S.V.M. and M. Narasimha Murty, 2002. SSVM: A simple SVM algorithm. Proceedings of the IEEE 2002 International Joint Conference on Neural Networks (IJCNN'02), 3: 2393-2398.