Using Linguistic Knowledge to Classify Non-functional Requirements in SRS documents

Ishrar Hussain, Leila Kosseim, and Olga Ormandjieva

Department of Computer Science and Software Engineering,
Concordia University, Montreal, Quebec, Canada
{h_hussa,kosseim,ormandj}@cse.concordia.ca

Abstract. Non-functional Requirements (NFRs) such as software quality attributes, software design constraints and software interface requirements hold crucial information about the constraints on the software system under development and its behavior. NFRs are subjective in nature and have a broad impact on the system as a whole. Being distinct from Functional Requirements (FR), NFRs are dealt with special attention, as they play an integral role during software modeling and development. However, since Software Requirements Specification (SRS) documents, in practice, are written in natural language, solely holding the perspectives of the clients, the documents end up with FR and NFR statements mixed together in the same paragraphs. It is, therefore, left upon the software analysts to classify and separate them manually. The research, presented in this paper, aims to automate the process of detecting NFR sentences by using a text classifier equipped with a part-of-speech (POS) tagger. The results reported in this paper outperform the recent work [6] in the field, and achieved a higher accuracy of 98.56% in the critical conditions of using 10-folds-cross-validation over the same data used by [6].The research reported in this paper is part of a larger project aimed at applying Natural Language Processing techniques in Software Requirements Engineering.

1. Introduction

Software Requirements Specification (SRS) document is one of the most important artifacts produced during the software development lifecycle. An SRS document specifies features of a software that enriches the understanding of the developers on the particular software. SRS documents are composed of two distinct kinds of requirements specification statements: (1) Functional Requirements, and (2) Non-functional Requirements.

Functional Requirements (FRs) are the sentences and propositions that specify the behavior of the software and/or software components, describing possible inputs and events of the environment and the corresponding required outputs. On the other hand, Non-functional requirements (NFRs) describe how the software system will provide the means to perform functional tasks; for example, software quality attributes software design constraints and software interface requirements [10]. Software quality NFRs include performance, reliability, maintainability, portability, robustness, security and likewise many others. Consider the following example:

"The System shall allow generation of Inventory Quantity Adjustment documents on demand. The System shall not require additional third party licenses resulting in royalty fees."[6]

Here, the first sentence describes the behavior of a system, and, therefore, is a functional requirement. In contrast, the second sentence explains a required quality of the system, and, therefore, is a non-functional requirement. Similar to FRs, NFRs also play a crucial role in SRS, because they can guide the developers to a completely alternative solution in the future design and implementation phase, because of the constraints they hold [4]. During requirements elicitation and analysis, NFRs tend to be stated in terms of either the qualities of the functional tasks or the constraints on them, which are expressed as functional requirements, as the former affect the semantics of the latter. In practice, the initial versions of both FRs and NFRs are written by the stakeholders with these FR and NFR sentences mixed together within the same paragraphs, and it often becomes hard for human analysts to factor out all the NFRs from the greater number of FRs. Empirical reports consistently indicate that neglecting NFRs in the requirements elicitation and analysis phase leads to project failures, or at least to considerable delays, and, consequently, to significant increases in the development final cost [2]. Thus, automating the task of detecting NFRs can reduce the risks associated with neglecting NFRs in the development process and therefore provide an assessment to the software analysts.

In this paper, we present our approach towards an effective method for automatic classification of textual requirements into two categories, namely, FRs and NFRs, by means of using a text classifier equipped with a part-of-speech (POS) tagger. Since the characteristics of FR and NFR remain within the scope of sentences, the classifier works only at the sentence-level.

The remainder of this paper is organized as follows: section 2 presents the related work. The methodology for automatic assessment of NFRs is introduced in section 3. Section 4 presents a discussion of the results and observations. Finally, our conclusions and directions for future work are outlined in section 5.

2. Related Work

The current processes to extract NFRs from SRS documents mostly rely on manual inspection, where an analyst reads the texts to identify a sentence manually as FR or NFR following different approaches (e.g. [5,7,8]). Research in this field to automate the process of extracting NFRs from SRS documents has been scarce.

A recent study of Cleland-Huang *et al.* [6] explored the use of text classification as an attempt to classify requirements statements into FR and NFR. As reported in their paper, their work attained a recall measure of 0.767 and a precision measure of 0.248 with their corpus. The authors used a stemmer to stem the words of the documents, and then selected keywords based on their high probability of occurrences in NFR statements. Their system then classified a statement as NFR, if the density of those selected keywords in that statement exceeds a particular threshold, else, otherwise.

The research work presented in this paper used the same corpus that was used by [6] and compares the performance of the resultant classifier with that of theirs. The

results reported in this work outperform the recent work in this field [6], and attain a higher accuracy of 98.56% using 10-folds-cross-validation over the data used by [6].

3. Methodology

Ideally, classifying the requirements as FR or NFR should be performed, or at least assisted, automatically. This section introduces our methodology aimed at improving the NFRs detection in requirements documents.

3.1 The Corpus

The corpus used by [6] was made freely available for download via [1]. It contains 15 SRS problem statements, all from different domains, with a total of 765 sentences: 495 (65%) of them were annotated as "NFR", while 270 (35%) of them as "FR". These will be referred to as *CorpusN* and *CorpusF*, respectively.

According to [6], all these statements were manually annotated by fifteen graduate students of DePaul University (who also work in the software industry as professionals). The same corpus is used in this project to train our classifier and also for testing its performance.

3.2. Syntactic Features

By definition of NFR, it can be realized that some categories of words are better indicators of NFR by their occurrences in the sentences. For example, NFR sentences often explain quality attributes of a component or the system as a whole, and such sentences are likely to contain Adjectives and Adverbs in them. Again, NFR sentences that explain constraints of the system are likely to contain Cardinals or numeric figures. Following these characteristics of NFR, as described in [4], we were motivated to choose a list of syntactic features as candidates and test their probabilities of occurrence in our collection NFR sentences (*CorpusN*), and thus, validate them to the most representative list of syntactic features.

We used the Stanford Parser [11] (equipped with Brill's POS tagger [3] and a morphological stemmer) to morphologically stem[1] the words and extract five syntactic features from each of the training instances (sentences) of the corpus. These features are:

- Number of Adjectives
  (*e.g. "good", "bad", "efficient" etc.*)
- Number of Adverbs
  (*e.g. "very", "well", "properly" etc.*)

[1] The morphological stemmer comes built-in with the Stanford parser [11]. It stems with the prior knowledge of the morphology of a word, and stems only to the point at which it retains its original POS class.

- Number of Adverbs that modify Verbs
  (*e.g. "well", "efficiently", "perfectly" etc.*)
- Number of Cardinals
  (*e.g. "1.56", "800x600", "twelve", "11" etc.*)
- Number of Degree Adjectives/Adverbs
  (*e.g. "better", "worse", "best", "more", "most" etc.*)

We only identified these features as candidates that can have some influence in the process of classifying a statement as NFR. Among these features, the ones that are valid for detecting NFR were to be selected automatically based on their ranks of the higher probabilities of their occurrences in NFR sentences of the training dataset. This probability measure for a candidate feature f_i is computed as follows:

$$\Pr(f_i) = \frac{\text{Frequency of } f_i \text{ in CorpusN}}{(\text{Frequency of } f_i \text{ in CorpusN} + \text{Frequency of } f_i \text{ in CorpusF})} \quad (\mathbf{1})$$

Note that this probability is not normalized, while 61% of the total number words in both the corpora belonged to CorpusN. The probability values of all the syntactic features are shown in Table 1.

Table 1. Probability Ranking of Syntactic Features (the numbers, computed by formula (1), indicates the probability of a feature to appear in Non-functional Requirements (i.e. *CorpusN*)

Feature	Probability
Cardinals	0.8762
Degree Adjectives/Adverbs	0.8571
Adverbs that modify Verbs	0.8571
Cutoff Threshold = 0.8	
Adverbs	0.7387
Adjectives	0.6193

Here, we manually selected a cutoff threshold (>0.8, in this case), and all the features exceeding the cutoff threshold were selected as valid (or most discriminating) features.

3.3. Keyword Features

Previous work of [6,13] has shown that NFR statements are mostly identifiable by the use of specific keywords that belong to different part-of-speech categories. The work of [6] also identified specific keywords, but with no regards to their parts-of-speech group. This allowed many words of unwanted parts-of-speech group to be included in their final list.

Analyzing the types of most probable words used in NFR, as described in [4], we have considered keywords of 9 different parts-of-speech groups separately— the frequencies of each becoming a feature in our final feature list. These 9 part-of-speech based keyword groups are:

– Adjective-keywords (coded as: *JJ_kw*)
– Adverb-keywords (coded as: *RB_kw*)
– Modal-keywords (coded as: *MD_kw*)
– Determiner-keywords (coded as: *DT_kw*)
– Verb-keywords (coded as: *VB_kw*)
– Preposition-keywords (coded as: *IN_kw*)
– Common Noun-keywords that appeared in singular form (coded as: *NN_kw*)
– Common Noun-keywords that appeared in plural form (coded as: *NNS_kw*)

Similarly to ranking the features, two different probability measures have been considered for ranking the keywords in each of the groups: (a) Unsmoothed Probability Measure, and (b) Smoothed Probability Measure.

3.3.1. Unsmoothed Probability Measure (UPM). Unlike [6], I considered taking the probability measure of a keyword occurring in a particular parts-of-speech (POS) category.

$$\Pr(Word, POS) = \frac{\text{Frequency of} < Word, POS > \text{in CorpusN}}{(\text{Frequency of} < Word, POS > \text{in CorpusN} + \text{Frequency of} < Word, POS > \text{in CorpusF})} \quad (2)$$

Keywords of a POS category were then ranked according to the higher values of unsmoothed probability measure. A high cutoff threshold is individually set for each group to select the most discriminating keywords that attain a higher value than the threshold,

3.3.2. Smoothed Probability Measure (SPM). While examining the values of the Unsmoothed Probability Measure, we found that some keywords, simply by chance, appeared only a few times in CorpusN, but never in CorpusF. This led them to have the highest probability value of 1, and was ranked on the top. On the other hand, some very discriminating keywords appeared in CorpusN many times, and also, by chance, appeared only a few times in CorpusF. UPM method ranks these keywords below the earlier ones, since they attain a probability value less than 1.

Also, with UPM, the keyword that appear, for example, once in CorpusN and never in CorpusF, and the keyword that appear, for example, 10 times in CorpusN, and never in CorpusF — both attain the same probability value of 1, and therefore, ranked together in the same place. But, clearly the second keyword is more discriminating than the first one.

To address these issues, we used a second measure that adds a small smoothing factor to all values of UPM. This factor takes into consideration how many more times the keyword in a particular POS group actually appears in CorpusN, than that of CorpusF. That is,

$$\text{Smoothed Pr}(Word, POS) = \Pr(Word, POS) + \frac{\text{Frequency of} < Word, POS > \text{in CorpusN}}{(\text{Frequency of} < Word, POS > \text{in CorpusF} + 1) \times \alpha} \quad (3)$$

Here, the constant α determines how much scaling one would like to add to the UPM value (the smaller the value of α, the higher the scaling). In our experiments, we used $\alpha = 10$.

Like UPM, keywords are also ranked according to their values of SPM, and a cut-off threshold is chosen to select the most discriminating keywords.

We used both UPM and SPM values to rank the keywords in their respective POS groups. Thus, we had the options, not only to manipulate the cutoff thresholds, but also to choose the best probability measures for each of the keyword groups. Table 2 illustrates the benefit of using SPM by showing an example from our corpora set.

Table 2. Choosing between UPM and SPM (here, ranking of keywords are based on SPM).

Keyword	POS	SPM	UPM	Frequency in CorpusN	Frequency in CorpusF
"*every*"	DT	1.741	0.941	16	1
"*no*"	DT	1.482	0.882	30	4
*Cutoff threshold set on **SPM*** = 1.3					
"*this*"	DT	1.225	0.875	7	1
"*these*"	DT	1.100	**1.000**	1	0

Table 2 shows the frequency values of the words "every" and "no", indicating that they are the most discriminating words in this category. It can also be guessed by understanding the nature of NFRs, which are most likely to contain descriptions of constraints that usually have determiner words, e.g. "every" and "no", as quantifiers. On the other hand, determiners like "this" and "these" simply appeared more in NFRs by chance. We find that, in this case, SPM successfully isolates the two most discriminating keywords of POS group DT[2], while UPM failed to create a proper ranking by setting its highest value (1.0) for the keyword "these". Therefore, here, we chose to rank the keywords by SPM values and set a cutoff threshold on them to select the most discriminating keywords. Thus, while training the system, for each of the keyword groups, either SPM or UPM can be chosen, along with the cut-off threshold to fine-tune the results. A slice of the list of the keywords selected in this way is shown in Table 3.

Table 3. Some of the keywords of different POS group, selected automatically by the keyword extractor program using both SPM and UPM set with different cut-off thesholds.

JJ_kw	RB_kw	MD_kw	DT_kw	VB_kw	IN_kw	NN_kw	NNS_kw
acceptable	only	should	every	accommodate	within	abuse	answer
active	prior	may	no	accomplish	than	accordance	aspect
ad-hoc	adequately	might		achieve	between	animation	attack
additional	appropriately			activate	during	appearance	audience
adjacent	approximately			adhere	about	attention	button
african	currently			agree	alongside	auditor	calculation
appealing	especially			appeal	per	avoidance	condition
...	...			...	...	...	...

[2] The Penn-tree bank tags are used for the keyword groups. DT signifies Determiners.

3.4. Feature Extraction and Classification

Our final list of features, therefore, is as follows:

- Number of Cardinals
- Number of Degree Adjectives/Adverbs
- Number of Adverbs that modify Verbs
- Adjective-keywords (coded as: *JJ_kw*)
- Adverb-keywords (coded as: *RB_kw*)
- Modal-keywords (coded as: *MD_kw*)
- Determiner-keywords (coded as: *DT_kw*)
- Verb-keywords (coded as: *VB_kw*)
- Preposition-keywords (coded as: *IN_kw*)
- Common Noun-keywords that appeared in singular form (coded as: *NN_kw*)
- Common Noun-keywords that appeared in plural form (coded as: *NNS_kw*)

To classify the sentences, we developed a Java-based feature extraction program that parses the sentences from the corpora, and extracts the values of all the features mentioned above, and uses Weka [14] to train C4.5 decision tree learning algorithm [12]. We used the implementation of C4.5 (revision 8) that comes with Weka (as J48), setting its parameter for the minimum number of instances allowed in a leaf to 6 to counter possible chances of over-fitting. The results are discussed in the next section.

4. Results and Analysis

The results came out to be exceptionally well when using the whole dataset for training and testing. Since the dataset was not very large, we also used 10-fold-crossvalidation, and the results were very good as well. Table 4 shows the summary of the results.

Table 4. Summary of the results

	Scheme	Correctly Classified Sentences	Incorrectly Classified Sentences	Kappa	Comment
Concatenation of CorpusN & CorpusF (Size = 475 + 270 = 765)	Training + Testing on same set	760 (99.35%)	5 (7.63%)	0.9856	Tree is of desirable characteristics, not sparse, and also not flat. None of the branches are wrongly directed.
	Cross-validation (10 Folds)	754 (98.56%)	11 (1.44%)	0.9682	

The resultant decision tree after training on the complete dataset also came out well-formed. The tree is shown in Figure 3.

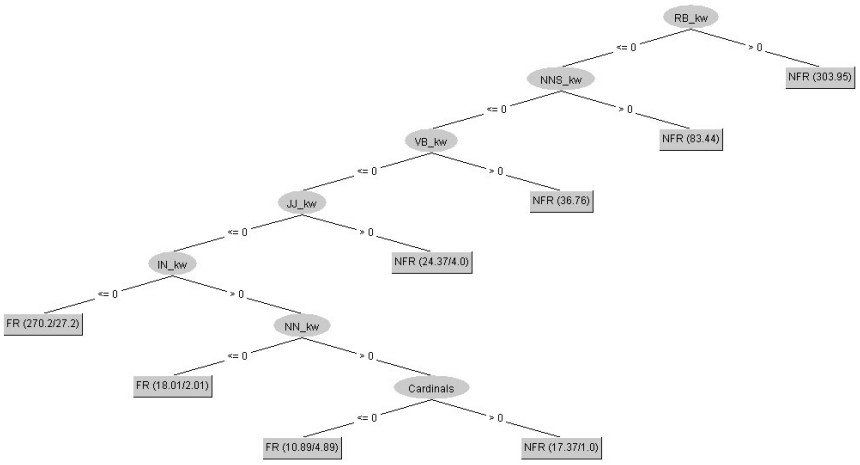

Fig. 1. The resultant C4.5 decision tree after training with the complete dataset.

Detailed results of using 10-fold-crossvalidation with the final confusion matrix and standard deviation on all measures are shown in Table 5 and 6 respectively.

Table 5. Confusion matrix when using 10-fold-crossvalidation

	Classified as	
	FR	**NFR**
FR	259	11
NFR	0	495

Table 5 shows that the retrieved all NFRs successfully (100% recall), without showing any sign of false negatives. Table 6, on the other hand, shows very low standard deviation over all the measurements taken during the iterations of crossvalidation (0.02 for precision and 0 for recall). This indicates that the results are likely to be robust.

Table 6. Detailed results of using 10-fold-crossvalidation

Test no.	Number of Training Instances	Number of Testing Instances	Number of Correctly Classified Instances	Number of Incorrectly Classified Instances	Total % of Correct	Total % of Incorrect	Kappa	Mean Absolute Error	RMS	For Classifying NFR			Characteristics of C4.5 Decision Tree		
										Precision	Recall	F-Measure	Total Nodes	Total Leaves	Total Rules
1	688	77	75	2	97.40	2.60	0.94	0.12	0.20	0.96	1.00	0.98	17	9	9
2	688	77	77	0	100.00	0.00	1.00	0.06	0.09	1.00	1.00	1.00	15	8	8
3	688	77	76	1	98.70	1.30	0.97	0.09	0.16	0.98	1.00	0.99	13	7	7
4	688	77	76	1	98.70	1.30	0.97	0.09	0.16	0.98	1.00	0.99	13	7	7
5	688	77	76	1	98.70	1.30	0.97	0.09	0.15	0.98	1.00	0.99	13	7	7
6	689	76	76	0	100.00	0.00	1.00	0.10	0.14	1.00	1.00	1.00	15	8	8
7	689	76	75	1	98.68	1.32	0.97	0.08	0.16	0.98	1.00	0.99	15	8	8
8	689	76	72	4	94.74	5.26	0.88	0.13	0.24	0.92	1.00	0.96	13	7	7
9	689	76	75	1	98.68	1.32	0.97	0.08	0.15	0.98	1.00	0.99	15	8	8
10	689	76	76	0	100.00	0.00	1.00	0.08	0.13	1.00	1.00	1.00	13	7	7
Mean			75.40	1.10	98.56	1.44	0.97	0.09	0.16	0.98	1.00	0.99	14.20	7.60	7.60
Standard Deviation(+/-)			1.35	1.20	1.57	1.57	0.04	0.02	0.04	0.02	0.00	0.01	1.40	0.70	0.70

Figure 2 and 3 shows similar phenomena, where the curves hardly experienced any drastic change.

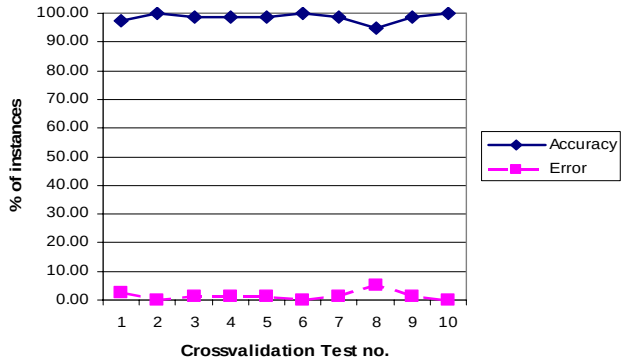

Fig. 2. Classifier's Accuracy/Error curve during 10-fold-crossvalidation

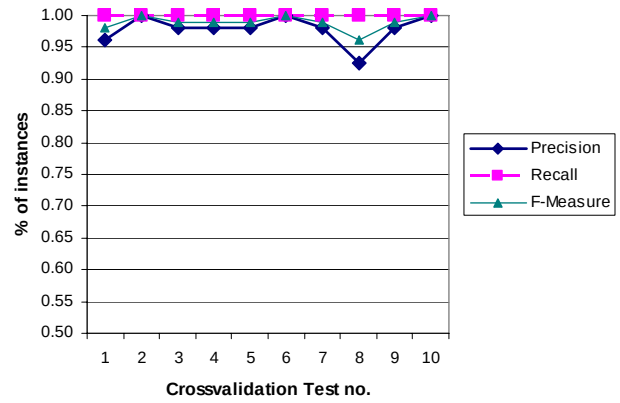

Fig. 3. Classifier's Precision/Recall/F-Measure curve for detecting NFR, during 10-fold-crossvalidation

Table 7 compares our results of precision and recall using 10-fold-crossvalidation to the results obtained by the previous work [6]. Here, we have further broken down our results into steps of improvement, i.e. firstly using syntactic features only, then using keyword features only, and then using both types of features, as documented previously.

Table 7. Comparison of our results to that of the previous work [6].

		Results	
		Precision	**Recall**
Work of Cleland-Huang *et al.* [6] - classification by density of keywords in the text		0.248	0.767
Our work: - classification by C4.5 decision tree learnner (10-fold-crossvalidation)	using Syntactic features Only	0.950	1.0
	using Keyword features Only	0.974	1.0
	using both Syntactic and Keyword features	0.978	1.0

The results of Table 7 show significant improvement over the most recent work [6] in the field. The classifier yielded high accuracy in performance, demonstrating results with 98.56% accuracy in the critical conditions of using 10-fold-crossvalidation. The precision and recall achieved by the classifier with 10-fold-cross-validation are 0.978 and 1.0 respectively, outperforming the work of [6] which attained the precision and recall as low as 0.248 and 0.767 respectively. Also, by using the syntactic features exclusively, and also by using our keyword features (that we selected based on their part-of-speech category) exclusively, yelled results which also surpass the performance of the work of [6] by a large margin.

5. Conclusions and Future Work

In this paper, a methodology for automatic classification of requirements by means of using a text classifier was presented. Our work extends the idea of [6] of using of Information Retrieval for classifying Non-functional requirements, and showed that using linguistic knowledge can help perform very well in classification task. Ours research aimed at assisting the software analysts in highlighting the NFRs in the users' textual SRS documents to avoid their further oversight in the development process, which can lead to poor quality of the final product and eventually to project failure.

The research reported in this paper is a part of a larger NLP-driven Requirements Engineering project which intends at using Natural Language Processing techniques in Requirements Engineering [9]. The goal of the work presented here was to increase the quality of the requirements text by deriving a module for the aforesaid project that would explicitly flag the requirements statements into FR and NFR for further processing. The module presented here can also be run exclusively as a stand-alone program to perform the classification task on requirements text. Our future work includes introducing more training and testing data, implementing a full-fledged prototype to demonstrate its use and a complete integration in our NLP-driven Requirements Engineering project.

Acknowledgements. We acknowledge Dr. Jane Cleland-Huang, co-author of [8], for making the corpus available online. We are also grateful to Dr. Doina Precup, for her valuable suggestions and feedback on the project.

References

1. Boetticher, G., Menzies, T., & Ostrand, T. (2007). PROMISE Repository of empirical software engineering data (http://promisedata.org/ repository), West Virginia University, Department of Computer Science, 2007 (Last retrieved: December 10, 2007)
2. Breitman, K., Leite, J. & Finkelstein, A. (1999). "The world's a stage: a survey of requirements engineering using a real-life case study". In *Journal of the Brazilian Computer Society*, 6 (1). pp. 13-37.
3. Brill, E. (1992). "A simple rule-based part-of-speech tagger". In *Proceedings of the 3rd Conference on Applied Natural Language Processing (ANLP-92)*, Trenton, Italy, April 1–3, pp. 152–155.
4. Chung, L., Nixon, B.A., Yu, E., & Mylopoulos, J. (2000). *Non-functional Requirements in Software Engiineering*, Kluwer Academic Publishers, 2000.
5. Chung, L. & Sapakkul, S. (2006). "Capturing and Reusing Functional and Non-functional Requirements Knowledge: A Goal-Object Pattern Approach," In *Proceedings of 2006 IEEE International Conference on Information Reuse and Integration*, Sept. 2006, pp.539-544.
6. Cleland-Huang, J., Settimi, R., Zou, X., & Solc, P. (2006). "The Detection and Classification of Non-Functional Requirements with Application to Early Aspects," In *Proceedings of 14th IEEE International Requirements Engineering Conference 2006 (RE'06)*, pp.36-45.
7. Cysneiros, L.M. & Leite, J.C.S. (2002) "Non-functional requirements: from elicitation to modelling languages," In *Proceedings of the 24rd International Conference on Software Engineering, 2002 (ICSE 2002)*. pp. 699-700.
8. Hill, R., Jun Wang & Nahrstedt, K. (2004) "Quantifying Non-functional Requirements: A Process Oriented Approach," Requirements Engineering Conference, 2004. In *Proceedings of 12th IEEE International*, 6-11 Sept. 2004, pp. 352-353.
9. Hussain, I., Ormandjieva, O., & Kosseim, L. (2007). "Automatic Quality Assessment of SRS Text by Means of a Decision-Tree-Based Text Classifier". In *Proceedings of the Seventh International Conference on Quality Software (QSIC 2007)*, pp. 209-218.
10. IEEE (1998). *IEEE recommended practice for software requirements specifications (IEEE Std 830-1998)*, 20 October, 1998, ISBN 0-7381-0332-2, New York: The Institute of Electrical and Electronics Engineers, Inc.
11. Klein, D., & Manning, C. D. (2003). "Accurate Unlexicalized Parsing". In *Proceedings of the 41st Meeting of the Association for Computational Linguistics (ACL'03)*, pp. 423-430, 2003.
12. Quinlan, J.R. (1993). *C4.5: Programs for machine learning. San Mateo*, CA: Morgan Kaufmann.
13. Rosenhainer, L. (2004). "Identifying Crosscutting Concerns in Requirements Specifications", Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design, 2004, Vancouver, Canada, Oct. 2004.
14. Witten, I. H., & Frank, E. (2005). *Data mining: Practical machine learning tools and techniques* (2nd ed.). San Francisco, CA: Morgan Kaufman.