IPTEK The Journal of Technology and Science, 31(3), 2020 (e/pISSN:2088-2033, 0853-4098)

DOI: 10.12962/j20882033.v31i3.7606

ORIGINAL RESEARCH

ALGORITHMS COMPARISON FOR NON-REQUIREMENTS CLASSIFICATION USING THE SEMANTIC FEATURE OF SOFTWARE REQUIREMENT STATEMENTS

Achmad An'im Fahmi | Daniel Siahaan*

[1]Dept. of Informatics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia

Correspondence

*Daniel Siahaan, Dept of Informatics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia. Email: daniel@if.ts.ac.id

Present Address

Gedung Teknik Informatika, Jl. Teknik Kimia, Surabaya 60111, Indonesia

Abstract

Noise in a Software Requirements Specification (SRS) is an irrelevant requirements statement or a non-requirements statement. This can be confusing to the reader and can have negative repercussions in later stages of software development. This study proposes a classification model to detect the second type of noise, the non-requirements statement. The classification model that is built is based on the semantic features of the non-requirements statement. This research also compares the five best-supervised machine learning methods to date, which are support vector machine (SVM), naïve Bayes (NB), random forest (RF), k-nearest neighbor (kNN), and Decision Tree. This comparison aimed to determine which method can produce the best non-requirements classification, model. The comparison shows that the best model is produced by the SVM method with an average accuracy of 0.96. The most significant features in this non-requirement classification model are the requirements statement or non-requirements, id statement, normalized mean value, standard deviation value, similarity variant value, standard deviation normalization value, maximum normalized value, similarity variant normalization value, value Bad NN, mean value, number of sentences, bad VB score, and project id.

KEYWORDS:

Irrelevant Requirements Noise, Non-Requirements Statement, Requirements Specifications, Statement, SVM

1 | INTRODUCTION

The Software Quality Assurance (SQA) process is considered very important to prevent the increase in costs required when developing software, especially at the requirements specification stage[1, 2]. Meyer emphasized that the Software Requirements Specification (SRS) process, which uses natural language, has several weaknesses compared to formal languages. Seven common mistakes cause these weaknesses that software developers make. Meyer introduced seven errors in software requirements that usually occur in software requirements specifications[1, 3–6]. The seven sins are noise, silence, overspecification, contradiction, ambiguity, forward reference, and wishful thinking. Several previous works focus on detecting the presence of seven errors in the software requirements specification[3, 5–7]. Another work by Enda and Siahaan focuses on correcting ambiguous requirements statements[4]. However, there have been no prior publications on noise detection in software requirements specifications from past research. Noise is not only a common problem in signal processing but also in the software requirements specification.

The given facts above are the motivation to research effective ways to detect noise in the software requirements specification. Noise is one of the types of errors that requirements engineers often make[1]. Noise may appear in the requirements specification process in Software Requirements Specifications (SRS). Noise occurs when the software developer includes information that is not relevant to the overall software requirements or includes non-requirements in the requirements specification. This can be confusing to the reader and can have negative repercussions in later stages of software development.

Noise is divided into two types, which are irrelevant requirements and non-requirement statements. Figure 1 shows the two types of noise. Irrelevant requirements discuss more topics outside the general topic that is being discussed by all relevant requirements. Non-requirements statements tend to contain topics that are also contained in the requirement statements from the same SRS. Although noise in various SRSs has different parts of the spoken order, non-requirement statements tend to have specific terms (such as project, schedule, object-oriented, and user acceptance).

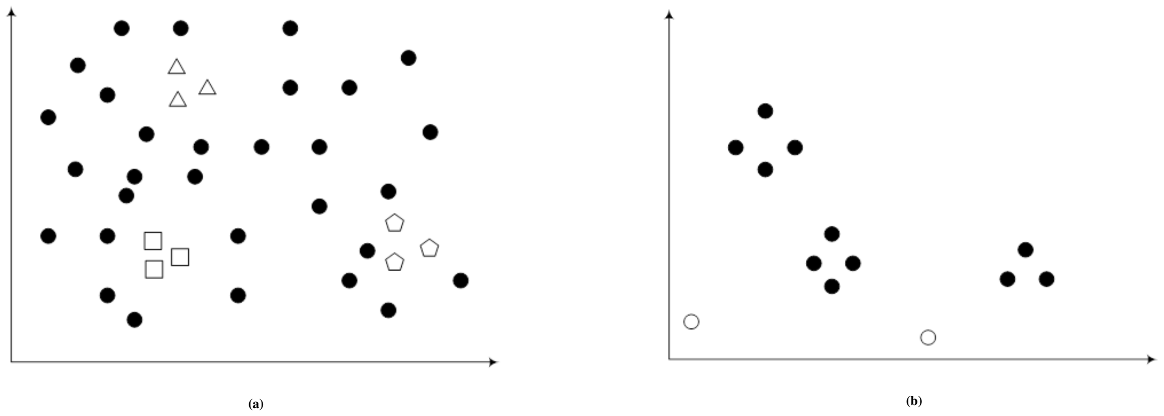


FIGURE 1 Illustration of non-requirements statements (a), illustration of irrelevant statements (b).

The existence of noise within software requirements specification could be the effect of the tendency of the engineer to say more than necessary. In the end, it could exhaust the project resource and thread the success of the project. Therefore, noise in software requirements specification should be detected and removed as early as possible. Noise detection in previous studies resulted in an unsatisfactory Kappa coefficient value, which is 0.4426. This is because the method developed in that study only detects the first type of noise, which is irrelevant requirement statements[8]. The author will improve the previous method by developing a classification model for detecting the second type of noise, non-requirement statements.

There are several studies on noise or outlier detection from text data. Several studies focus on adapting unsupervised machine learning to detect noise[9–13]. Another study focused on detecting irregularities in documents using conceptual charts[14]. The study provides a graphic visualization of the deviations that occur in text data.

This study exercises five well-known representable supervised methods, i.e., Support Vector Machine (SVM), Naïve Bayes (NB), Random Forest (RF), k-Nearest Neighbor (kNN), and Decision Tree[15–18]. The contribution of this research is to build a classification model for the non-need statement for waiting for noise.

Furthermore, a classification model will be made based on the results of the best classification method. Thus, the non-requirements classification modeling is expected to improve the quality of the SRS.

2 | MATERIAL AND METHOD

2.1 | Dataset

This study used 648 software requirements statements that were extracted manually from 14 SRS documents. Table 1 shows the dataset used in this study. Each document refers to a different project from various problem domains[8]. Five annotators label each requirements statement as requirements, non-requirements, or irrelevant. The label is a value of boolean type. The annotator will label the statement with 1 (true) in the noise column if and only if it thinks the statement is noise. The annotator will label the status with 0 (false) in the noise column if and only if it thinks the statement is a relevant requirements statement.

TABLE 1 The requirements specification dataset.

ID	Project Name	Requirements	Non-Requirements
DA-1	Submit Job	13	0
DA-2	System Administrator	17	0
DA-3	Archive Administrator	39	0
DA-4	SPG Application	6	1
DA-5	System Administrator Staff	33	0
DA-6	Software Development	65	28
DA-7	Display System	106	9
DA-8	Internet Access	64	4
DA-9	Meeting Initiator	33	0
DA-10	Online System	17	0
DA-11	Library System	86	11
DA-12	IMSETY System	70	0
DA-13	Manage Student Information	24	1
DA-14	PHP Project	75	3
Total		**648**	**57**

Measuring the reliability between annotators on the non-requirements statement shown by the red diagram in Figure 6, the average reliability between annotators is 0.87. This shows that the five annotators have a very good level of reliability.

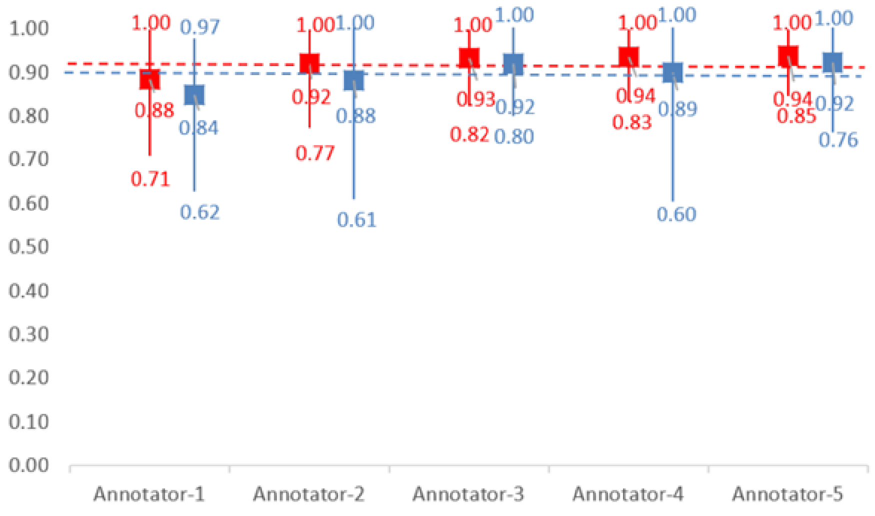


FIGURE 2 Reliability between annotators on a non-requirements statement.

2.2 | Development of a Non-Requirements Statement Classification Model

The steps for building a classification model for non-requirements statements can be seen in Figure 2. This step consists of three main processes: creating a sentence-level feature, a discourse-level feature, and a noise classification model.

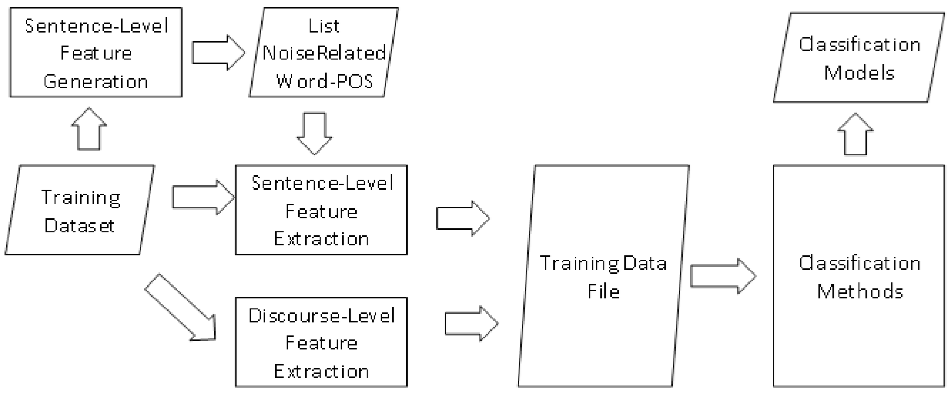

Sentence-Level Feature Generation
List NoiseRelated Word-POS
Classification Models
Training Dataset
Sentence-Level Feature Extraction
Training Data File
Classification Methods
Discourse-Level Feature Extraction

FIGURE 3 Building the classification models.

2.3 | Sentences Level Feature Generation

As shown in Figure 3, sentence-level feature generation comprises two main processes, i.e., sentence preprocessing and feature generation. Sentence processing involves part-of-speech (POS) tagging, lemmatization, stopword removal, and word-POS tag frequency counter. Based on the study carried out by Hussain[19], we considered seven word-pos tag pairs to be counted, i.e., word-MD (modal), word-NN (noun), word-VB (verb), word-RB (adverb), word-JJ (adjective), word-DT (determinant), and word-other (conjunction, preposition, cardinal, etc.). Since NN tags (e.g., NN, NNS, NNP, and NNPS), we grouped them into NN tags. The same approach was applied to other POS tags. We maintained a bag of words for each type of POS tag. Each bag of words contains unique word-POS tags. Based on the seven bags of words, the Semantic-based Word-POS tags frequency counter counts the number of word-POS tag occurrences in non-requirement and requirement statements for each type of word-POS tag pairs.

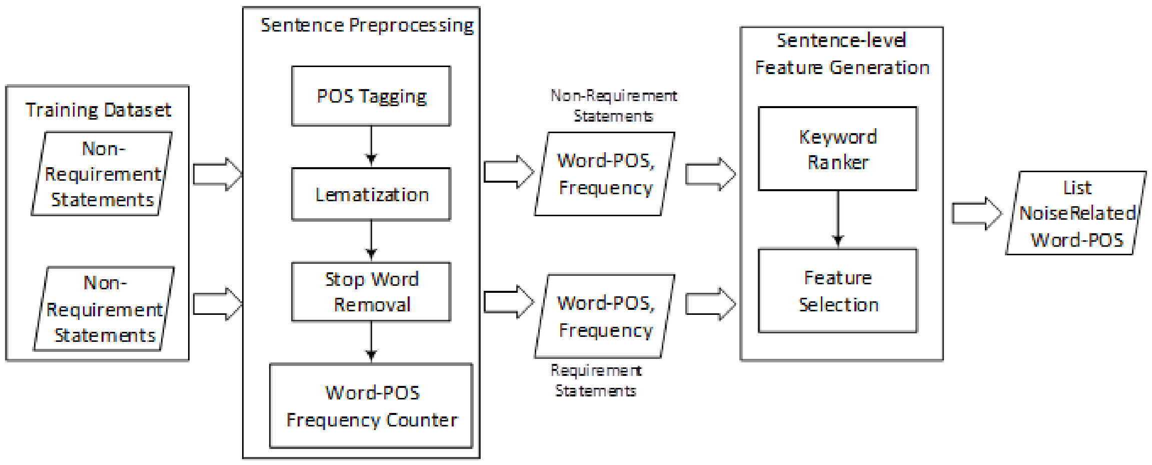


FIGURE 4 Automatic noise-related sentence-level feature generation.

Sentence-level feature generation involves two main processes, i.e., keyword ranker and feature selection. Keyword ranker uses the output from the previous process to calculate the strength of discrimination (in terms of likelihood ratio or LR) of each word-POS tag. LR of a word-POS can be calculated using equation 1. Keyword ranker incrementally orders the results.

$$LR(word - POS) = \frac{\textit{frequency of (word-POS)in Corpus Non-Requirements}}{\textit{frequency of (word-POS)in both corpus}} \tag{1}$$

2.4 | Feature Extraction

The next process was extracting sentence-level and discourse-level features from the training data. This training data was used to build the classification model for detecting the non-requirement type of noises. Figure 4 and Figure 5 show the process diagram for extracting sentence-level and discourse-level features, respectively. In sentence-level feature extraction, the frequency counter process refers to the seven bags of word-POS generated from the previous process. Since the corpus contains a limited number of vocabularies, we need to extend the scope to handle broader cross-domain projects. We used Wordnet thesaurus to find similar words given a similarity threshold.

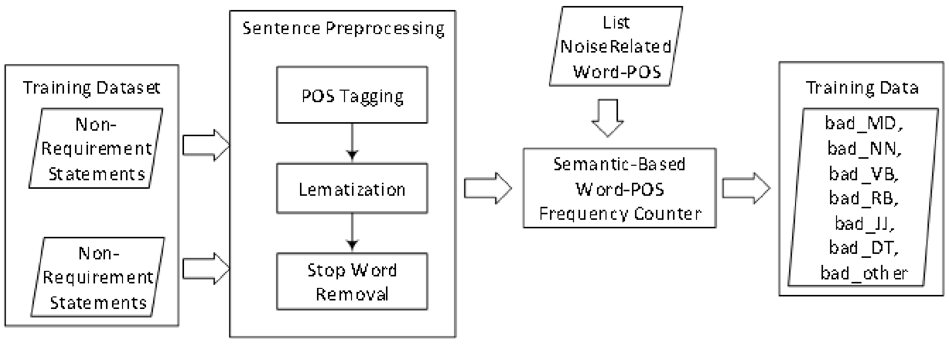

Training Dataset
Non-Requirement Statements
Non-Requirement Statements
Sentence Preprocessing
POS Tagging
Lematization
Stop Word Removal
List NoiseRelated Word-POS
Semantic-Based Word-POS Frequency Counter
Training Data
bad_MD, bad_NN, bad_VB, bad_RB, bad_JJ, bad_DT, bad_other

FIGURE 5 Sentence-level feature extraction.

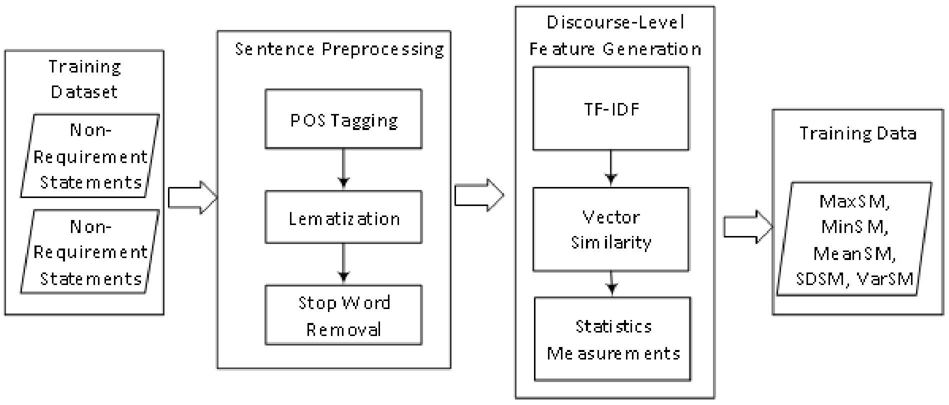


FIGURE 6 Discourse-level feature extraction.

3 | RESULTS AND DISCUSSION

3.1 | Test Results

The features that have been made in the previous chapter were tested using the weka application. The initial process is cross-validation of all project IDs. Furthermore, training and testing methods were used in a cycle, namely: the DA-1 project ID was used as test data, and the DA-2 to DA-14 project IDs were used as training data. DA-2 project IDs are used as test data, and project IDs DA-1, DA-3 to DA-14 as training data and so on so that DA-14 project IDs are test data and project IDs DA-1 through DA-13 are training data.

Table 2 shows the performance of each classification method in a cross-validation environment. Table 3 shows the performance of each method in building a classification model for non-requirements statements and testing each dataset. From the training and testing process, it can be seen that SVM and Random Forest are almost relative inaccurate (with a variance of 0.012 and

0.011). The accuracy of the kNN and the Decision Tree is below it, with a variance of 0.016. Meanwhile, Naive Bayes has the highest variance value, which means that its accuracy value is the lowest than the other four methods. These results indicate that the training model produced by SVM has a higher average accuracy value (0.883) than the other four methods. The classification model produced by SVM also has the highest average accuracy value (0.910). These results are consistent with research by [15, 20], wherein in terms of accuracy, SVM can produce better classification results than other classification methods.

TABLE 2 The accuracy of classification models in the training process.

ID	SVM	Naive Bayes	Random Forest	kNN	Decission Tree
DA-1	0.880	0.712	0.869	0.820	0.847
DA-2	0.880	0.719	0.872	0.815	0.861
DA-3	0.877	0.700	0.865	0.813	0.860
DA-4	0.882	0.707	0.863	0.812	0.855
DA-5	0.880	0.741	0.875	0.834	0.868
DA-6	0.916	0.794	0.904	0.856	0.902
DA-7	0.884	0.731	0.865	0.815	0.865
DA-8	0.884	0.693	0.872	0.822	0.846
DA-9	0.878	0.750	0.863	0.813	0.855
DA-10	0.883	0.705	0.870	0.818	0.856
DA-11	0.881	0.705	0.861	0.827	0.849
DA-12	0.870	0.690	0.851	0.806	0.836
DA-13	0.883	0.713	0.865	0.825	0.865
DA-14	0.883	0.691	0.869	0.839	0.867
μ	**0.883**	0.718	0.869	0.823	0.860

Table 3 also shows statistical information from the classification results of each method. This confirms that, in general, the classification model produced by SVM has a more stable performance with respect to the population size of the dataset and the ratio between requirements and non-requirements statements. The precision-recall plot in Figure 7 and the ROC-recall plot in Figure 8 of each method in two-dimensional space can measure the classification ability to distinguish the two possible outcomes. The classification model that can produce the highest number of results located above the diagonal line is considered the best model. Two graphical plots show that the classification model produced by SVM has higher potential uses than the others. In addition, for the variance of classification results among problem domains, this study also showed an insignificant variance for all methods.

TABLE 3 The accuracy of classification models in the testing process.

ID	SVM	Naive Bayes	Random Forest	kNN	Decission Tree
DA-1	1,000	0,462	0,923	0,846	0,923
DA-2	1,000	0,529	0,941	0,941	0,941
DA-3	0,974	0,974	0,974	0,974	0,974
DA-4	1,000	0,333	1,000	1,000	1,000
DA-5	0,939	0,121	0,758	0,697	0,606
DA-6	0,585	0,662	0,600	0,646	0,585
DA-7	0,877	0,868	0,877	0,868	0,877
DA-8	0,875	0,813	0,891	0,766	0,875
DA-9	0,970	0,970	0,970	0,970	0,970
DA-10	0,882	0,588	0,824	0,706	0,941
DA-11	0,895	0,616	0,895	0,709	0,884
DA-12	0,986	0,829	0,971	0,900	0,957
DA-13	0,875	0,833	0,875	0,833	0,875
DA-14	0,880	0,507	0,867	0,653	0,827
μ	**0,910**	0,650	0,883	0,822	0,874
σ^2	0,012	0,062	**0,011**	0,016	0,016
σ_X	0,107	0,248	**0,105**	0,125	0,128

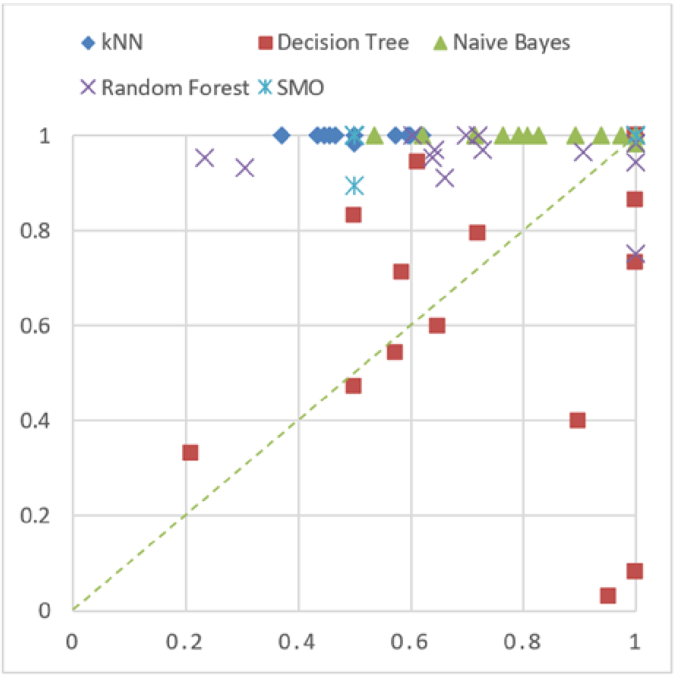


FIGURE 7 ROC-recall plots of five classification methods.

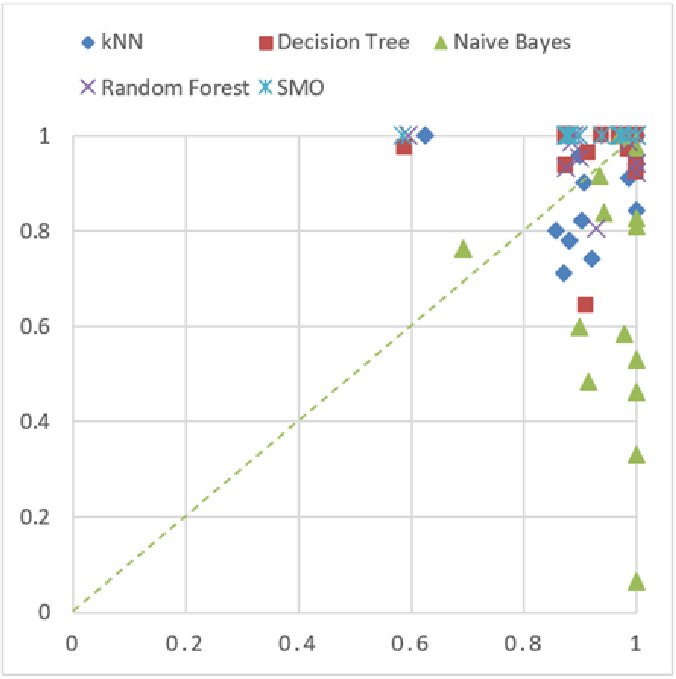


FIGURE 8 Plot precision-recall of five classification methods.

3.2 | SVM Representation

Simple representation of SVM as in equation 3.

$$min_{f\xi_i} \|f\|_k^2 + C \sum_{i=1}^{l} \xi_i y_i f(x_i) \geq 1 = \xi_i, for\ all\ i\ \rightarrow \xi_i \geq 0 \tag{2}$$

The variable ξ_i is called the slack variable to measure the error made at point x_i, y_i). Kernel functions play an important role in SVM performance. It is based on reproducing the Hilbert Kernel space as in Eq. 4.

$$K(x, x') = \left(\Phi(x), \Phi(x')\right) \tag{3}$$

If K is a positively symmetric definite function, which satisfies Mercer's Rule,

$$K(x, x') = \sum_{m}^{\infty} a_m \, \Phi_m(x) \, \Phi_m(x'), a_m \geq 0 \tag{4}$$

$$\int\int K(x, x') \, g(x) \, g(x') \, dxdx' > 0, \; g \in L_2 \tag{5}$$

The polynomial kernel is a popular method for non-linear modeling. The second kernel is usually preferred because it avoids problems with hessian being zero.

$$K(x, x') = (x, x')^d \tag{6}$$

$$K(x, x') = (< x, x' > +1)^d \tag{7}$$

3.3 | Attribute Selection

Attribute selection is made to find out which attribute is the most significant. The attribute evaluator uses InfoGainAttributeEval and the Ranker lookup method from the weka application. Table 4 explains the title code in Table 5. In Table 5, it can be seen that all features can affect accuracy (with the greatest variance of 0.0001). From the attributes selection results, it can be seen that of the nineteen features used to build the model, only eleven features that are considered significant classify non-requirement statements. The eleven features represent semantic and statistical features.

TABLE 4 Description of the title code on the name of attributes.

Title Code	Attribute Name	Information
a	maxValNorm	Maximum normalized value
b	meanValNorm	Mean normalized value
c	STDValNorm	The normalized standard deviation value
d	varSMNorm	Variant normalization value
e	maxVal	Maximum value
f	meanVal	Mean value
g	STDVal	Standard deviation value
h	varSM	Variant value
i	bad_NN	Bad NN value
j	bad_VB	Bad VB value
k	bad_punctuation	Other bad value in words

3.4 | Pearson Correlation Coefficient

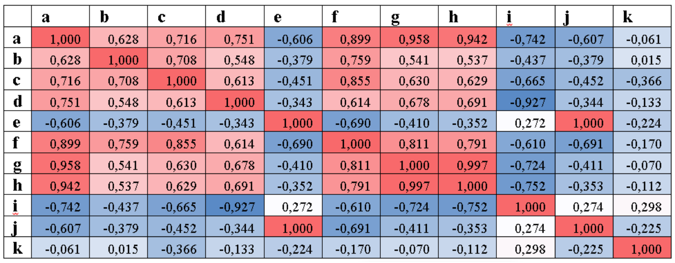
Table 6 shows the correlation between variables after feature selection. Explanation of letters a-k in the top and left side of the title as described in Table 4.

3.5 | Discussion

From the above research results, it can be seen that there is a set of semantic features that can classify requirement statements and non-requirement statements. Meanwhile, SVM was chosen because it is the best-supervised method for classifying requirement statements and non-requirement statements.

TABLE 5 Feature selection results with the treatment of a zero-value feature on every deleted dataset.

ID	a	b	c	d	e	f	g	h	i	j	k
DA-1	0.072	0.022	0.020	0.021	0.005	0.057	0.075	0.071	0.000	0.010	0.011
DA-2	0.071	0.021	0.019	0.020	0.005	0.056	0.074	0.071	0.000	0.010	0.000
DA-3	0.066	0.020	0.021	0.019	0.005	0.050	0.069	0.065	0.000	0.010	0.013
DA-4	0.071	0.019	0.017	0.018	0.005	0.054	0.075	0.072	0.000	0.010	0.012
DA-5	0.087	0.021	0.019	0.020	0.005	0.060	0.092	0.088	0.000	0.010	0.013
DA-6	0.044	0.000	0.000	0.000	0.006	0.036	0.051	0.047	0.016	0.012	0.015
DA-7	0.064	0.000	0.000	0.019	0.006	0.039	0.071	0.067	0.000	0.012	0.015
DA-8	0.071	0.000	0.019	0.016	0.005	0.050	0.076	0.072	0.000	0.011	0.000
DA-9	0.070	0.021	0.021	0.018	0.005	0.051	0.073	0.070	0.000	0.010	0.013
DA-10	0.073	0.022	0.020	0.021	0.005	0.054	0.072	0.069	0.000	0.010	0.000
DA-11	0.074	0.022	0.021	0.014	0.005	0.058	0.081	0.078	0.000	0.011	0.000
DA-12	0.069	0.020	0.018	0.018	0.005	0.055	0.072	0.069	0.000	0.010	0.000
DA-13	0.064	0.000	0.018	0.016	0.005	0.050	0.069	0.065	0.000	0.010	0.000
DA-14	0.086	0.022	0.023	0.020	0.000	0.067	0.083	0.077	0.000	0.000	0.015
μ	0.070	0.015	0.017	0.017	0.005	0.053	0.074	0.070	0.001	0.010	0.008
σ^2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
σ_X	0.010	0.010	0.007	0.005	0.001	0.008	0.009	0.009	0.004	0.003	0.007



	a	b	c	d	e	f	g	h	i	j	k
a	1,000	0,628	0,716	0,751	-0,606	0,899	0,958	0,942	-0,742	-0,607	-0,061
b	0,628	1,000	0,708	0,548	-0,379	0,759	0,541	0,537	-0,437	-0,379	0,015
c	0,716	0,708	1,000	0,613	-0,451	0,855	0,630	0,629	-0,665	-0,452	-0,366
d	0,751	0,548	0,613	1,000	-0,343	0,614	0,678	0,691	-0,927	-0,344	-0,133
e	-0,606	-0,379	-0,451	-0,343	1,000	-0,690	-0,410	-0,352	0,272	1,000	-0,224
f	0,899	0,759	0,855	0,614	-0,690	1,000	0,811	0,791	-0,610	-0,691	-0,170
g	0,958	0,541	0,630	0,678	-0,410	0,811	1,000	0,997	-0,724	-0,411	-0,070
h	0,942	0,537	0,629	0,691	-0,352	0,791	0,997	1,000	-0,752	-0,353	-0,112
i	-0,742	-0,437	-0,665	-0,927	0,272	-0,610	-0,724	-0,752	1,000	0,274	0,298
j	-0,607	-0,379	-0,452	-0,344	1,000	-0,691	-0,411	-0,353	0,274	1,000	-0,225
k	-0,061	0,015	-0,366	-0,133	-0,224	-0,170	-0,070	-0,112	0,298	-0,225	1,000

FIGURE 9 Pearson correlation coefficient.

4 | CONCLUSION

The detection of non-requirements statements on the SRS can be done by classifying the statements in the SRS. From the classification process of five methods, SVM, Naive Bayes, Random Forest, kNN, and Decision Tree, the SVM method is the best method for detecting non-requirements statements indicated by an average accuracy value of 0.96. Apart from that, from the precision-recall and ROC-recall plots, the points generated by the SVM method are above the diagonal line compared to other methods with the x and y axes approaching the value 1.

The features that affect the results of noise detection are maximum normalization value, mean normalization value, standardized standard deviation value, variant normalization value, maximum value, mean value, standard deviation value, variance value, bad NN value, bad VB value, and other bad grades. The suggestion that can be given for further development is using other methods to get a better score. In addition, it is hoped that the development of a classification model in the Naive Bayes method is expected so that this method can detect non-requirements statements better.

References

1. Meyer B. On Formalism in Specifications. IEEE Software 1985 jan;2(1):6–26. http://ieeexplore.ieee.org/document/1695257/.

2. Romano S, Scanniello G, Fucci D, Juristo N, Turhan B. The effect of noise on software engineers' performance. In: Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement New York, NY, USA: ACM; 2018. p. 1–10. https://dl.acm.org/doi/10.1145/3239235.3240496.

3. Purnomo W, Siahaan DO. Pendeteksian Overspesification Pada Dokumen Spesifikasi Kebutuhan Perangkat Lunak. Inspiration : Jurnal Teknologi Informasi dan Komunikasi 2017 jun;7(1):1–9. https://jurnal.akba.ac.id/index.php/inspiration/

article/view/2431.

4. Enda D, Siahaan D. Rekomendasi Perbaikan Pernyataan Kebutuhan yang Rancu dalam Spesifikasi Kebutuhan Perangkat Lunak Menggunakan Teknik Berbasis Aturan. Jurnal Teknologi Informasi dan Ilmu Komputer 2018 may;5(2):207. http://jtiik.ub.ac.id/index.php/jtiik/article/view/627.
5. Sahadi FVS, Siahaan DO, Yuhana UL. Pendeteksian Istilah Berbeda Pada Dokumen Spesifikasi Kebutuhan Perangkat Lunak (Skpl). SCAN - Jurnal Teknologi Informasi dan Komunikasi 2015;10(3):9–16. http://www.ejournal.upnjatim.ac.id/index.php/scan/article/view/619.
6. Siahaan D, Umami I. Natural Language Processing for Detecting Forward Reference in a Document. IPTEK The Journal for Technology and Science 2012 nov;23(4). http://iptek.its.ac.id/index.php/jts/article/view/99.
7. Yang H, de Roeck A, Gervasi V, Willis A, Nuseibeh B. Analysing anaphoric ambiguity in natural language requirements. Requirements Engineering 2011 sep;16(3):163–189. http://link.springer.com/10.1007/s00766-011-0119-y.
8. Manek PG, Siahaan D. Noise Detection in Software Requirements Specification Document Using Spectral Clustering. JUTI: Jurnal Ilmiah Teknologi Informasi 2019 mar;17(1):30–37. http://juti.if.its.ac.id/index.php/juti/article/view/771.
9. Cai X, Zhang R, Gao D, Li W. Simultaneous Clustering and Noise Detection for Theme-based Summarization. Proceedings of 5th International Joint Conference on Natural Language Processing 2011;p. 491–499. http://aclweb.org/anthology/I11-1055.
10. Jiang Sy, An Qb. Clustering-Based Outlier Detection Method. In: 2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery IEEE; 2008. p. 429–433. http://ieeexplore.ieee.org/document/4666153/.
11. Pamula R, Deka JK, Nandi S. An Outlier Detection Method Based on Clustering. In: 2011 Second International Conference on Emerging Applications of Information Technology IEEE; 2011. p. 253–256. http://ieeexplore.ieee.org/document/5734938/.
12. Gan G, Ng MKP. k -means clustering with outlier removal. Pattern Recognition Letters 2017 apr;90:8–14. https://linkinghub.elsevier.com/retrieve/pii/S0167865517300740.
13. Mahapatra A, Srivastava N, Srivastava J. Contextual Anomaly Detection in Text Data. Algorithms 2012 oct;5(4):469–489. http://www.mdpi.com/1999-4893/5/4/469.
14. Kamaruddin SS, Hamdan AR, Bakar AA, Mat Nor F. Deviation detection in text using conceptual graph interchange format and error tolerance dissimilarity function. Intelligent Data Analysis 2012 may;16(3):487–511. https://www.medra.org/servlet/aliasResolver?alias=iospress{&}doi=10.3233/IDA-2012-0535.
15. Liu Z, Lv X, Liu K, Shi S. Study on SVM Compared with the other Text Classification Methods. In: 2010 Second International Workshop on Education Technology and Computer Science, vol. 1 IEEE; 2010. p. 219–222. http://ieeexplore.ieee.org/document/5459006/.
16. Colditz RR. An evaluation of different training sample allocation schemes for discrete and continuous land cover classification using decision tree-based algorithms. Remote Sensing 2015;7(8):9655–9681.
17. Setyawan DA, Fatichah C. Enhancement of Decission Tree Method Based on Hierarchical Clustering and Dispersion Ratio. JUTI: Jurnal Ilmiah Teknologi Informasi 2020 jul;18(2):179–187.
18. Hakim L, Rochimah S, Fatichah C. Klasifikasi Kebutuhan Non-fungsional Menggunakan FSKNN Berbasis ISO/IECC 25010. JUTI: Jurnal Ilmiah Teknologi Informasi 2019 aug;17(2):107–116.
19. Hussain I, Ormandjieva O, Kosseim L. Automatic Quality Assessment of SRS Text by Means of a Decision-Tree-Based Text Classifier. In: International Conference on Quality Software; 2007. p. 209–218.
20. Amancio DR, Comin CH, Casanova D, Travieso G, Bruno OM, Rodrigues FA, et al. A Systematic Comparison of Supervised Classifiers. PLoS ONE 2014 apr;9(4):1–14. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.

0094137.

How to cite this article: Fahmi A.A., Siahaan D., (2020), Algorithms Comparison for Non-Requirements Classification Using The Semantic Feature of Software Requirement Statements, *IPTEK The Journal of Technology and Science*, *31(3):343–352*.