Local Feature Extraction for Noise Detection in Software Requirements Specification Document

Ahmad Mustofa[1, a], Daniel Siahaan[1,b]

[1]Informatics Department, Institut Teknologi Sepuluh Nopember, Indonesia

[a]mustofa.ahmad12@mhs.if.its.ac.id, [b]daniel@if.its.ac.id

Keywords: software requirements, local features, noise detection.

Abstract. Requirements specification is the first step of a software development cycle. If errors occurred in this step, errors would automatically occur in the next step. Errors in software requirements consist of noise, ambiguous, conflict, and inconsistency. This paper introduces a classification model for automatically detecting noises in Software Requirement Specification (SRS) document. The model was build based on local features extracted from each requirement statements within an SRS document. These local features are the statistical properties calculated by measuring the similarities between requirement statements within the same SRS document. Then, the set of data from the document were combined with other set of data from different documents to make a single dataset. This dataset was then used to build a classification model. The best model was obtained using Support Vector Machine (SVM) as classification method. Its performance was relatively high, i.e. 82.96% accuracy, 73.17% true positive rate, 84.07% true negative rate, and 46.51% F1 measure.

Introduction

Detecting errors in software requirements holds an important role in software development since the first step in software development is requirement specification. The sooner an error is detected, the cheaper cost is needed to correct. In [1] the authors represented error in software requirement into 7 groups, renowned as Meyer's seven sins. Noise, one of Meyer's seven sin, is an error that caused by the presence of an element in the text that does not carry information relevant to any feature of the problem. There are 2 types of noise, redundancy and remorse. Remorse is defined as a requirement that holds irrelevant information to the problem domain, while redundancy is defined as a requirement that holds exactly same information with another requirement. Type of noise that will be used in this paper is remorse.

In past several years, machine learning has made a rapid progress in text processing. Bag-of-word (BOW) is a common approach to represent a text document as a vector of unique term/word [2]. Each column of the vector holds the weight value of corresponding term. In [2, 3] the authors use TF-iDF (Term Frequency-invers Document Frequency) as term weighting method to categorize text document. TF-iDF is also used in [4] to classify an email is either spam or not. Depends on these facts, TF-iDF will also be used in this paper as term weighting method.

Noise in Software Requirement Specification (SRS) document can only be recognized locally and since we treated a single requirement as a single document, a global Bag of Words (BOW) approach is not applicable. In this study, we proposed a statistical approach to represent a single requirement so then it can be used to classify a requirement is either a noise or not.

The remainder of this paper is organized as follows. In section 2, we elaborate the method used to detect a noise in SRS document. Section 3 presents the performance evaluation results. Finally, we conclude the paper in section 4.

Noise Detection Model

To perform noise detection in requirement statements of SRS document, we need to classify each requirement statement is either noise or not. The most important issue in requirement statement classification is how to represent each requirement statement. A famous BOW approach is not applicable since noises in SRS document can only be recognized locally.

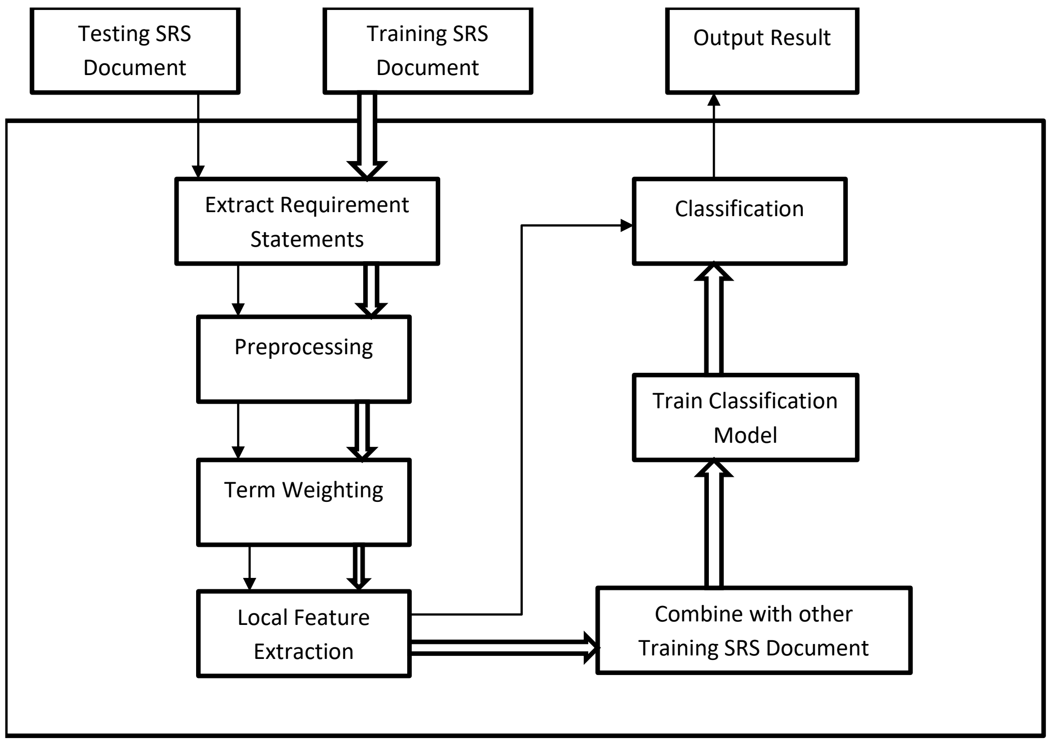


Fig 1 The architecture of noise detection system

The first step to detect noises in SRS document is extracting requirement statements. These requirement statements are then used to build noise detection model. Noise detection model is built by doing 4 important steps as follows: data preprocessing, terms weighting, local features extraction, and train the classification model.

Data Preprocessing

Each requirement extracted from an SRS document is preprocessed to get a vector of unique term needed in Term Weighting process. There are 3 main steps in preprocessing text i.e. stopword and punctuation removal, tokenizing, and stemming. In stopword and punctuation removal, we remove all the conjunction and non-alphabetical element from the text. The output of this process is then tokenized in tokenizing process. In tokenizing step, we split the text into some unique token / terms. Then these terms are stemmed to get stem of each unique term.

Term Weighting

After being preprocessed, each requirement statement of an SRS document is represented by a vector of unique terms. These terms are combined to get the Bag of Words (BOW) of the SRS document. This BOW is then used to build weight vectors of each requirement statement.

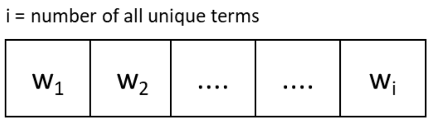


Fig 2 Weight vector

To get the weight of term i (w_i) from a requirement statement can be calculated as [3] :

$$w_i = tf_i \times log_2 \frac{N}{df_i} \tag{1}$$

where tf_i is the frequency of the term in the corresponding requirement statement, N is the number of requirement statements in the corresponding SRS document, and df_i is number of requirement statement in the corresponding SRS document that contain term i.

Local Feature Extraction

Before combined with other requirement statements from other SRS documents, we need to extract the local feature of each requirement statement of an SRS document. This local feature is obtained by calculating the similarity between requirement statements of the same SRS document. Similarity of requirement statement i and requirement j $s_{i,j}$ is calculated as [5, 6]:

$$s_{i,j} = \frac{\sum_{k=1}^{n} w_{i,k} \times w_{j,k}}{\sqrt{\sum_{k=1}^{n} w_{i,k}}^{2} \times \sqrt{\sum_{k=1}^{n} w_{j,k}}^{2}} \quad (2)$$

where n is number of unique terms in the corresponding SRS document, $w_{i,k}$ is weight of term k in requirement statement i, and $w_{j,k}$ is weight of term k in requirement statement j.

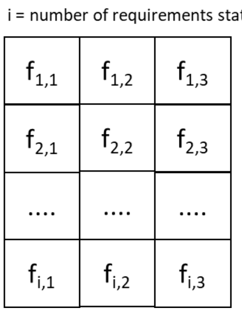
Statistical features have been used to extract local feature of an image to access it's quality [7]. Based on this research, we used 3 statistical properties to represent a requirement statement. These 3 features of a requirement statement i are calculated as follows:

$$f_{i,1} = \frac{\sum_{k=1,k\neq i}^{N} s_{i,k}}{N-1} \quad (3)$$

$$f_{i,2} = max_{k \neq i, k \in N} s_{i,k} \quad (4)$$

$$f_{i,3} = \sqrt{\frac{\sum_{k=1,k\neq i}^{N} s_{i,k} \times f_{i,1}}{N-1}} \quad (5)$$

where N is the number of requirement statements in the corresponding SRS document. The result of this step is a feature matrix that will be used later to train the classification model.



i = number of requirements statement

$f_{1,1}$	$f_{1,2}$	$f_{1,3}$
$f_{2,1}$	$f_{2,2}$	$f_{2,3}$
....	....	....
$f_{i,1}$	$f_{i,2}$	$f_{i,3}$

Fig 3 Feature matrix of an SRS document

Result and Analysis

Dataset

To measure the performance of our system, we use 10 SRS document with total 405 requirement statements. 41 requirement statements are labeled as noise and 364 requirement statements are labeled as normal. It shows that the data we used to train the classification model is not balance. It may produce a bad classification model that will lead to a bad performance of classification. To overcome this problem, we use Synthetic Minority Over-sampling Technique (SMOTE) to balance the training data. SMOTE is an oversampling algorithm that exploit the similarity of the minority data [8]. We use 10-fold cross validation in our experiments, 9 parts are used as training data and the remaining part is used as testing data.

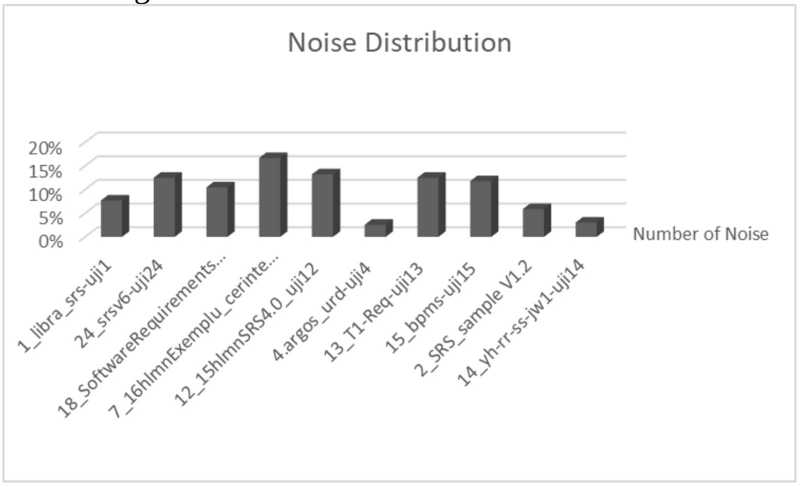


Fig 4 Noise distribution in dataset

Result and Analysis

We use 3 classification method i.e. Support Vector Machine (SVM), Neural Network, and K Nearest Neighbor (KNN) to test the performance of our proposed model. Each classification method has performance as follow.

Table 1 Performance of SVM, NN, and KNN

Classification Method	**Accuracy (%)**	**TP Rate (%)**	**TN Rate (%)**	**F1 Measure (%)**
SVM	82.96	73.17	84.07	46.51
Neural Netwok	89.63	9.76	98.63	16
K Nearest Neighbor	87.41	9.76	96.15	13.51

From Table 1, we can see that SVM has better performance in term of true positive rate and F1 measure. When the data is not balanced like in this case, F1 measure has an important role in determining performance of the system. The greater F1 measure of a system, the more balanced it is to classify the data correctly.

Using SVM as classification method, we try to classify all the requirement statements of all SRS document.

Table 2 Final Performance Result

SRS Document	**Accuracy (%)**	**TP Rate (%)**	**TN Rate (%)**	**F1 Measure (%)**
1_libra_srs-uji1	100	100	100	100
24_srsv6-uji24	91.67	100	90.48	75
18_SoftwareRequirements Specification-uji18	66.28	87.5	64.1	32.56
7_16hlmnExemplu_cerinte _software -uji7	100	100	100	100
12_15hlmnSRS4.0_uji12	89.62	92.86	89.13	70.27
4.argos_urd-uji4	97.44	0	100	NaN
13_T1-Req-uji13	71.88	37.5	76.79	25
15_bpms-uji15	70.59	50	73.33	28.57
2_SRS_sample V1.2	94.12	100	93.75	66.67
14_yh-rr-ss-jw1-uji14	96.97	0	100	NaN

In SRS 24_srsv6-uji24 that examine requirement specification of an Academic Information System, there are 2 normal requirement statements that classified as noise (false positive). These 2 requirement statements are "*The timeframe to upload data to the central database is the day that grades are due to the university and one week before the start of the new semester*" and "*All data transmitted to the central database will be encrypted*". These 2 requirement statements determined by the system as noise because they have low similarity in case of term frequency with other requirement statements in the same SRS document, but the experts labeled them as a normal requirement statement because they are semantically similar with other requirement statements in the same SRS document. On the other hand, there is 1 noise requirement that classified as normal (false negative) in SRS 18_SoftwareRequirementsSpecification-uji18 that examine requirement specification of an Online Library System of a University. This requirement, "*The university information security system must be compatible with the Internet applications*", has high similarity in case of term frequency with other requirement statements in the same document so that the system determined it as a normal requirement statement. The expert labeled it as a noise because it is outside the SRS document's context.

Conclusion and Future Works

In this paper, we proposed a noise detection model to detect noises in SRS documents automatically. Our proposed model achieved a promising result as initial research in this field. Even so, our proposed model failed to measure the semantical similarity between requirement statement. More experiments need to be done in the future to make use of semantic information of the requirement statement so that it can overcome the weakness of our proposed model.

References

[1] B. Meyer, "On Formalism in Specification," *IEEE Software,* 1985.

[2] Q. Luo, E. Chen and H. Xiong, "A semantic term weighting scheme for text categorization," *Expert Systems with Applications,* no. 38, p. 12708–12716, 2011.

[3] B. Trstenjak, S. Mikac and D. Donko, "KNN with TF-IDF Based Framework for Text Categorization," *Procedia Engineering,* no. 69, pp. 1356-1364, 2014.

[4] H. Xu and B. Yu, "Automatic thesaurus construction for spam filtering using revised back propagation neural network," *Expert Systems with Applications,* no. 37, pp. 18-23, 2010.

[5] G. Salton and M. J. McGill, Introduction to Modern Information Retrieval, New York, 1986.

[6] A. Singhal, "Modern Information Retrieval: A Brief Overview," Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 2001.

[7] W. Zhou, L. Yu, W. Qiu, Y. Zhou and M. Wu, " Local Gradient Patterns (LGP): An Effective Local-Statistical-Feature Extraction Scheme for No-Reference Image Quality Assessment," *Information Sciences,* Vols. 397-398, pp. 1-14, 2017.

[8] N. V. Chawla, K. W. Bowyer, L. O. Hall and P. W. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence,* no. 16, pp. 321-357, 2002.