Redundancy Detection Of Sentence Pairs In The Software Requirements Specification Documents With Semantic Approach

E D Oktaviyani 1*, Licantik 2
1,2 Universitas Palangka Raya, Kampus UPR Tunjunng Nyaho Jalan Yos Sudarso Palangka Raya (73112), Kalimantan Tengah, Indonesia
Email: *enny.obrien@gmail.com

Abstract. Software requirements specification document must complete, unambiguous and must correct because can lead to failure in software development. It must contain sentences not redundant. Meyer’s seven sins explain some errors in software requirements specification document, one of which is a Noise. Variant of the noise is redundancy. Redundancy is repeating the same information, but using different terms or phrases, thus giving the impression that new information.

This research to analyze performance redundancy detection in sentences pair on software requirement specification documents using WordNet based semantic similarity methods. This study tries to develop a framework that has been proposed in previous studies in different framework that is to detected redundancy in sentences without fact extraction. The proposed approach and evaluation process uses Kappa values to determine whether reliable framework that can be used to detect redundancy in sentence pairs by using experiments in two scenarios to get good results. Our experimental result show the performance of proposed framework can used to help reduce redundancy when creating software requirements specification documents and may contributes to enhance the quality of document. Shown with good kappa values and interpretation of the Kappa value is substantial agreement. It means many agreements between experts and framework.

1. Introduction
One of the documents produced by requirements engineering is a Software Requirements Specification (SRS) document. The characteristics of Software requirement specification documents or called SRS document are true, unambiguous, complete, consistent, ordered according to their importance or stability, can be verified, can be modified and can be traced [1]. SRS is a critical stage in SDL (Software Development Live Cycle), then in the making SRS documents must be careful in order to produce a tool quality software [2]. Errors that occur in the need engineering affect the failure of the product. No matter how good next stage [3]. Sentences Characteristics in the SRS document is the form of statement sentences. Incomplete, incorrect and ambiguous SRS documents can lead to failure in software development. Meyer’s seven sins explain some errors in SRS documents, one of which is a Noise. Variant of the noise is redundancy. Redundancy is repeating the same information, but using different terms or phrases, thus giving the impression that new information [4].

Many research have been conduct to analyze SRS documents. For example, [5] check whether written needs are valid or not, and examine ambiguities related to homonyms, heteronyms and homograph. The proposed method can work efficiently if the requirements in the specification document are no more than 6 words, for example a simple declarative sentence. [6] discuss importance
impact formal inspection from requirements specifications for Software Quality Assurance and a review of requirements specifications. [6] also discusses various errors and the possible causes of errors and the effects of those errors. Such errors include inaccuracies, ambiguities, functionality that is missing from the document requirements specification. While in research [7] describes an automated approach for detecting ambiguous software requirement specification. This paper study a set of nine classification algorithms from the machine learning community and evaluate which algorithm performs best to detect the ambiguous software requirement specification. The Research related to detecting redundancy is a study conducted by. [8] conduct research to detect and delete redundancy information using graph-based representation algorithms at the sub-snippet level on news documents. The weakness of the proposed method is that the approach to the pairing of snippets used is very limited and does not use synonyms or semantic analysis. In the previous studies [9], we have proposed a framework for detecting redundancy in facts pair in SRS documents. In[9], the pair of sentences entered will be cut according to the rules cutting the sentence into several clauses based on subordinating conjunction and coordinating conjunction. The similarity of phrases are calculated using WordNet based semantic similarity method. The results show the approach we are proposing has good performance and can be used to detect redundancy in SRS documents threshold values 0,58 and kappa value 0,747.

Our study is related to research [9] but using a different framework is to cut one stage, it is the fact extraction stage. In this study we propose a new framework for detecting redundancy in sentence pairs in SRS documents without fact extraction. We try to analyze framework performance whether it can work goodness to detect redundancy in sentence pairs although no using the fact extraction stage. The proposed framework consists of several processes, they are input sentences pair, preprocessing, calculate similarity value between sentences pair, determine threshold values, and determine redundancy or no redundancy. Finally, we try to analyze the performance of redundancy detection in sentence pairs. Testing and evaluating the framework using Kappa values and interpretation of kappa values to find out the strength of the proportion of agreements between solutions proposed with linguists. We hope this study can contribute to making better SRS documents and produce good quality SRS documents.

2. Method

Figure 1 show the proposed framework in this study. In figure 1 can be seen, the sentences pairs entered do not facts extraction stage. The proposed framework consists several steps, they are input of sentences pairs, preprocessing, calculating the sentences pairs semantic similarity, determine the threshold value, and last determine whether the sentences pairs redundancy or no redundancy based on threshold values. At this framework, the sentences pair not be through the fact extraction process

- **Method of collecting data**

  The dataset in this study consisted of 205 sentences pair taken from eight SRS documents. The dataset in this study same with previous study at [9] taken from selected SRS documents:

  1. User Requirement Document – Personal Information Filter versi 2.0.0
  2. User Requirement Document – The QIS Application urd-2.0.1874
  3. User Requirement Document - the IMSETY system Version 1.2
  4. User Requirement Document - Secure Digital Archiving versi 2.0.0
  5. Software Requirement Specification - Libra: An Economy-Driven Cluster Scheduler versi 1.0
  6. User Requirement Document – Interface Specification Tools – Calisto version 1.0.1
  7. User Requirement Document – The Archive – version 3.0
  8. Software Requirement Document – Web Publishing System Version 1.0

Figure 2 show an illustration of taking a sentences pair in a SRS document.
3. Results and Discussion
For example from figure 3, Step 1 input the same sentence as the previous framework from user requirement document Personal Information Filter version 2.0.0 page 7.

$K_1 = \text{“The PIF group consists of multiple teams, each team performs a specific task”}$

$K_2 = \text{“A team is a subset of the group”}$.
Step 2 is preprocessing that consists of process tokenize, stopwords removal, and stemming. The example of sentences preprocessing

**Tokenize:**
- K1: “The” “PIF” “group” “consists” “of” “multiple” “teams” “” “each” “team” 
  “performs” “a” “specific” “task”
- K2: “A” “team” “is” “a” “subset” “of” “the” “group”

**Stopwords removal:**
- K1: pif group consists multiple teams team performs specific task
- K2: team subset group

**Stemming:**
- K1: group consist multiple team team perform specific task
- K2: team subset group

Step 3 is calculate similarity value in sentence pairs K1 and K2. Similarity value in sentence pairs calculated using WordNet-based semantic similarity [9][10]. WordNet is a lexical database for English which is the result of research from Princeton University in the computational linguistic field. The lexical database is a data set that stores information on semantic relations between synonyms set (synset). Based on calculated with Simialrity values from sentences pairs Sim \((K_1, K_2) = 0.707\). Table 1 show semantic similarity matriks for sentences pairs K1 and K2.

**Tabel 1. Semantic Similarity Matriks For Sentences Pairs K1 And K2**

|     | K2     | team | subset | group |
|-----|--------|------|--------|-------|
| K1  | Group  | 0.89 | 0.8    | 1     |
|     | Consist| 0    | 0.29   | 0     |
|     | Multiple| 0.33 | 0.38   | 0.4   |
|     | Team   | 1    | 0.5    | 0.89  |
|     | Team   | 1    | 0.5    | 0.89  |
|     | Perform| 0.29 | 0      | 0.33  |
|     | Specific| 0.4  | 0.46   | 0.5   |
|     | task   | 0.38 | 0.43   | 0.46  |

Then calculated sentences similarity using fast heuristic method.

\[
Sim(K_1, K_2) = \frac{\Sigma(\text{max values of each rows}) + \Sigma(\text{max values of each columns})}{\Sigma(\text{number of rows and columns})}
\]  
\[
Sim(K_1, K_2) = \frac{2.8 + 4.98}{11}
\]

\[
Sim(K_1, K_2) = 0.707
\]

Step 4 is determine threshold values obtained from sentences similarity value. Threshold values determine using semantik similarity value of sentences pair from table 1. The scenario to determine the threshold value is by observing the similarity value of the sentences pairs in dataset then testing with several threshold values to obtain a high level of accuracy of the kappa value. Choosing the right threshold value will improve the accuracy and quality of the kappa value.

The last step is determining the sentences pairs is redundancy or not redundancy. If similarity values > threshold values, then the statements pair is redundancy. If similarity values < threshold values, then a not redundancy

To Test and evaluate in this study, we used Kappa values. Kappa values indicate the proportion of agreement between the two examiners, in this case an expert in English, to test the accuracy of the
data redundancy as the first test and the proposed framework as second test. Kappa values are used to determine whether the proposed framework is reliable. In this research, testing to detect redundancy in eight of SRS from 205 sentence pairs. To detect redundancy, we conducted experiments with two scenarios. The first scenario is to experiment with a threshold value of 0 to 1 to get the highest kappa value. From the results of the first scenario the highest kappa value is 0.749 at the threshold value of 0.7. Table 2 show The composition data to calculate kappa values for first scenario. From table 2, the kappa value will be calculated as follows

\[
p_o = \frac{a+d}{n}
\]

\[
p_o = \frac{44+141}{205} = 0.902
\]

\[
p_e = \left[\frac{n_1}{n} \times \frac{m_1}{n}\right] + \left[\frac{n_0}{n} \times \frac{m_0}{n}\right]
\]

\[
p_e = \left[\frac{51}{205} \times \frac{57}{205}\right] + \left[\frac{154}{205} \times \frac{148}{205}\right] = 0.612
\]

\[
Kappa, K = \frac{p_o - p_e}{1 - p_e}
\]

\[
Kappa, K = \frac{0.902 - 0.612}{1 - 0.612} = 0.749
\]

**Table 2.** The Composition Data to Calculate Kappa Value With a 0.7 Threshold Value In The First Scenario

| expert redundancy | Not redundancy | Total |
|-------------------|----------------|-------|
| redundancy        | 44 (a)         | 13 (b) | 57 (m1) |
| Not redundancy    | 7 (c)          | 141 (d) | 148 (m0) |
| Total             | 51 (n1)        | 154 (n0) | 205 (n) |

Then doing experiment with the second scenario to obtain more accurate data, the second scenario is done by reducing the threshold value data range, that is at the threshold of 0.66 to 0.75. The second scenario produces the highest kappa value is 0.777 at the threshold of 0.7. Table 3 show the composition data to calculate kappa values for second scenario

**Table 3.** The Composition Data to Calculate The Value Of Kappa Second Scenario For SentencesPair

| expert redundancy | Not redundancy | Total |
|-------------------|----------------|-------|
| redundancy        | 42 (a)         | 8 (b) | 50 (m1) |
| Not redundancy    | 9 (c)          | 146 (d) | 155 (m0) |
| Total             | 51 (n1)        | 154 (n0) | 205 (n) |

From table 3, the kappa value will be calculated as follows

\[
p_o = \frac{a+d}{n}
\]
\[ p_o = \frac{42 + 146}{205} = 0.917 \]

\[ p_e = \left[ \left( \frac{n_1}{n} \right) \times \left( \frac{m_1}{n} \right) \right] + \left[ \left( \frac{n_0}{n} \right) \times \left( \frac{m_0}{n} \right) \right] \]

\[ p_e = \left[ \left( \frac{51}{205} \right) \times \left( \frac{50}{205} \right) \right] + \left[ \left( \frac{154}{205} \right) \times \left( \frac{155}{205} \right) \right] = 0.629 \]

\[ Kappa, K = \frac{(p_o - p_e)}{(1 - p_e)} \tag{5} \]

\[ Kappa, K = \frac{0.917 - 0.629}{(1 - 0.629)} = 0.777 \]

The results of the first scenario with a threshold of 0 to 1 and the results of the second scenario with a threshold of 0.66 to 0.75 are shown in Table 4.

| First Scenario | Second Scenario |
|----------------|-----------------|
| Threshold      | Kappa           | Threshold | Kappa |
| 0.1            | 0.006           | 0.66      | 0.70  |
| 0.2            | 0.006           | 0.67      | 0.697 |
| 0.3            | 0.010           | 0.68      | 0.713 |
| 0.4            | 0.033           | 0.69      | 0.706 |
| 0.5            | 0.186           | 0.7       | 0.749 |
| 0.6            | 0.538           | 0.71      | 0.742 |
| **0.7**        | **0.749**       | **0.72**  | **0.754** |
| 0.8            | 0.687           | **0.73**  | **0.777** |
| 0.9            | 0.244           | 0.74      | 0.759 |
| 1              | 0.000           | 0.75      | 0.755 |

Figure 3 shows a graph of the experimental results in the first scenario, and Figure 4 shows a graph of the experimental results in the second scenario. In figure 3 shows that the highest kappa value is 0.749 at the threshold values is 0.7. To get a valid kappa value, then recalculate at the threshold point near to the highest kappa value shown in figure 4. The near points are taken from the threshold values 0.66 to 0.75. Based on this near point, the calculation is carried out in the second scenario.
Based on experiments in the first scenario and the second scenario, the performance of the framework for detecting redundancy in sentence pairs works very well at a threshold value of 0.73 with a kappa value of 0.777. The table 5 shows the interpretation of kappa values. Based on the interpretation of kappa values, performance of the framework is substantial agreement.
Tabel 5. The Interpretation of Kappa Values

| Kappa Index Values | Proportion Agreement       |
|--------------------|----------------------------|
| < 0                | less than chance agreement |
| 0.01 – 0.20        | slight agreement           |
| 0.21 – 0.40        | fair agreement             |
| 0.41 – 0.60        | moderate agreement         |
| 0.61 – 0.80        | substantial agreement      |
| 0.81 – 1           | almost perfect agreement   |

4. Conclusion
Our study aims to analyze the performance of detect redundancy in sentences pair SRS documents. This study tries to develop a framework that has been proposed in previous studies in different ways. In this study the dataset is a pair of sentences not through the fact extraction process. Based on experiments with two scenarios, it can be concluded that the performance of the framework proposed in this study can be used to detect redundancy in sentence pairs which can help reduce redundancy when creating software requirements specification documents and may contributes to enhance the quality of SRS. The threshold used 0.73 at the kappa value is 0.777 with the interpretation of the Kappa value being a substantial agreement. These results indicate that there are many agreements between experts and the framework.

Acknowledgment
We are grateful to the Informatic Department, University of Palangka Raya and fellow researchers who have supported our research.

References
[1] S. Engineering and S. Committee, 2011. IEEE recommended practice for software requirements specifications, 1998, 2011.
[2] R. Pressman, 2012. Rekayasa Perangkat Lunak 7th ed. Yogyakarta: Andi, Yogyakarta, 2012.
[3] S. Nurfauziah, 2017. Pendeteksian Ketidaklengkapan Kebutuhan Dengan Teknik Klasifikasi Pada Dokumen Spesifikasi Kebutuhan Perangkat Lunak, 9(2).
[4] B. Meyer, 1985. On Formalism in Specification,” IEEE Softw., 2(1), pp. 6–26.
[5] umber A and Bajwa I.S, 2011. No Title,” in Minimizing Ambiguity in Natural Language Software Requirements Specification, pp. 102–107.
[6] P. Gupta, 2010. Impact of formal Inspection to Software Requirements Specification For Software Quality Assurance,” Glob. J. Enterp. Inf. Syst., 2(2), pp. 11–19.
[7] M. H. Osman and M. F. Zaharin, 2018. Ambiguous software requirement specification detection: An automated approach,” Proc. - Int. Conf. Softw. Eng., no. July, pp. 33–40.
[8] K. Thadani and K. McKeown, 2008. A framework for identifying textual redundancy, Coling 2008 - 22nd Int. Conf. Comput. Linguist. Proc. Conf., 1, no. August, pp. 873–880.
[9] E. D. Oktaviyani and D. Oranova, 2013. Redundancy Detection Of Sentence Pairs In The Software Requirements Specification Documents With Semantic.
[10] S. Christina and E. D. Oktaviyani, 2017. Identifying the relevant page numbers that referred by the back-of-book index using syntactic similarity and semantic similarity, in 2017 Second International Conference on Informatics and Computing (ICIC), pp. 1–6.