Machine Learning Technique for Identifying Ambiguities of in Software Requirements

Prerana Chaithra[1], Dr. Shantharam Nayak[2]

[1]VTU Research Scholar, RVCE & Associate Professor, Information Science & Engineering, Sapthagiri College of Engineering, affiliated to VTU, Bengaluru
[2]Professor & Principal advisor, affiliated to Visveswaraya Technological University, Bengaluru
[1]preranachaithra15@gmail.com, [2]shantaram_nayak@yahoo.com

Article History: Received: 11 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 10 May 2021

Abstract

Generally most of the requirements are expressed in Natural Language. Walkthroughs, reviews and inspections are the methods currently used in the industries to identify ambiguities, inconsistencies. We have difficulties identifying ambiguities and have the tendency to overlook inconsistencies in large Natural Language requirements. A reviewer may ignore some errors while going through the requirements because he might assume that the first interpretation of the software requirements document that understood by him is the intended interpretation, unaware of the other possible understanding. He unconsciously disambiguates an ambiguous requirement. Currently most of the automation tools are in the nasal stage. Options are open for research as to how to reduce ambiguities in software requirements. The proposed method is to identify the number of ambiguities in software requirements during software analysis using Machine Learning, which in turn reduces errors and leads to a better product. It also helps to ease the job of the software analysts.

Keywords: Ambiguities, Machine Learning, Natural Language, Software Requirements

I. Introduction

Software have become a part of day to day life. The errors in the software lead to low quality software. To develop high quality software, the ambiguities in the software must be removed. The basic step is to start the removal of ambiguities in starting phase of software development. Removal of errors in the requirements analysis phase will reduce the propagation of errors to the later stages of the software development. These errors can be reduced by using various methods[14]. Unfortunately most of the methods are time consuming.

If there are ambiguities in the requirements, it leads to misinterpretation of the requirements and hence leads to low quality software products. Sometimes it may also lead to wrong products. Such requirements cause wrong design and wrong implementation of the product [2]. To develop high quality software these ambiguities must be reduced or removed. The software must satisfy the quality attributes [15].

2. Problem Statement

The software requirements are the ones which determine the software that should be developed. Software requirements analysis is the initial step in the software production. Requirements are collected from the stakeholders of the product; therefore, they are in natural language format. There are different methods for reviewing the contents of the software requirement. Most of these methods are manual[16]. Very few methods are automated but are in nasal state.

Lot of research must be done to reduce ambiguities in software requirements [12]. The motivation is to have high quality software in lesser time by having high quality software requirements. In this paper we have introduced a method to identify the ambiguities which are present in the requirements, so that the ambiguities can be removed to have good quality software requirements to develop the software.

3. Literature Survey

Understanding the software requirements plays a significant role the software development. The ambiguities in the software requirements need skilled Business analysts and hence they have to update their skillsets very often. [13].

The prototype developed by Ashima Rani [1] automatically detects different types of ambiguities. This algorithm detects different types of ambiguities. It is used to find the ambiguities of various types in the SRS.

The prototype developed by M. Q. Riaz [2] refers to the preprocessing stage that consists of finding ambiguous requirements in the Software Requirements Specification. It removes requirements which are ambiguous and leads to more suitable requirements. It cannot handle sentences which are Compound and complicated phrases in the natural language requirements.

The method suggested by Mohd Hafeez Osman [3] is finds ambiguous software requirements. Text mining and machine learning are used to remove ambiguity. Dataset is extracted by using Text mining. The training-set, is used by this approach to detect ambiguities in SRS. An approach was developed by them for finding ambiguous requirements.

The clarification given by Ali Olow Jim'ale Sabriye [4] that the requirement is said to be ambiguous if the requirement has multiple interpretation. Due to this misinterpretation the developers may develop a software which is not according to the customer's requirements. They have proposed approach for detecting syntax and semantic ambiguities in requirements. The ambiguities are detected by using Parts of Speech tagging technique. They have compared the human and system capabilities of detecting the syntactic ambiguity.

The methodology suggested by T. Hovorushchenko [5] evaluates the information of the SRS and shows how much information is sufficient to determine the quality of the SRS information. The requirement for developing methodology for sufficient evaluation of the SRS quality is shown in the result.

The approach given by R. Navarro-Almanza [7] explains that the requirements quality determines the software quality being developed as the Software Requirements depicts the detailed needs and expectations. Requirements classification is done by using Deep Learning. Using CNN for basis, an approach was proposed by them. The results are average on Software Requirements.

Analysis of quality indicators of the SRS was done by Azlin Nordin [8]. These quality factors are found in a way that they are good for automation. They have found quality of the requirements on the basis of sentence and also document. Suitable quality indicator was developed for these types. Next investigation was to find the way of documenting these quality indicator.

The system proposed by A. Takoshima [10] is based on the varieties in SRS that leads to systematic inspection. The two phases of inspection are as follows. External inspectors do the First phase of inspection. Second phase of inspection is done by the project team.

The prototype developed by Shalini Ghosh [11] called as ARSENAL with the aim of constructing a scalable, robust and trainable framework that bridges the gap formal tools and natural language requirements. The idea behind ARSENAL lies in automated generation of a model which is a formal model using requirements in the natural language format. This model is used for checking different types of ambiguities and also the traceability to connect implementations to requirements they implement.

4. Implementation

The implementation of the Software Requirements (SR) Parser shown in figure 1 consists of

- Installation of MongoDB Local Client and establishing connection with the Client
- Creating database and collection in MongoDB
- Parsing available SR documents with the Python NLTK kit and storing the keywords
- Removal of Duplicates from the database thereby having a Trained dataset (Master) using Machine Learning techniques

Step 1: Software requirements document is given as input to the SR Parser.
Step 2: If the SR document is complete then it will be send for document cleansing phase where the unwanted space will be removed else if it is incomplete then it will be rejected.
Step 3: The cleansed SR document will be parsed through the SR document parser
Step 4: Using Machine Learning Classification algorithm which is a supervised algorithm ambiguous and unambiguous words are classified.
Step 5: Check whether the keywords match with the ambiguous keywords print the number of matching and mismatching words.
Step 6: If the keywords match the ambiguous keywords in the database then based on that modify the SR document and the modified SR document will be checked again for the completeness and ambiguous words which leads to ambiguous requirements.
Step 7: If the ambiguous keywords are removed then we get the final SR document without ambiguous words.

In this way we can get SR document without ambiguous words.

The GUI developed processes the SR file and provides the list of matched entries and unmatched entries. In the background the engine searches for the keywords already fed to the system. These keywords indicate the quality parameters.

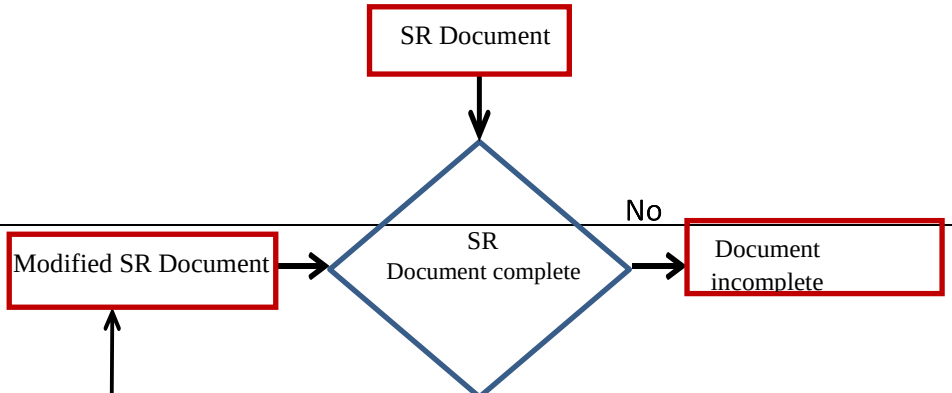

Figure 1. Flowchart of the developed prototype for identifying ambiguities in software requirements.

Figure 1 depicts the flowchart of the developed prototype for identifying ambiguities in Software Requirements (SR).

5. Result and Discussion

The Software Requirements(SR) document consists of the different varieties of requirements. They are functional-, nonfunctional-, user- and system- requirements. Therefore this document must be of high quality to produce high quality software. A quality SR document exhibits the following characteristics: Unambiguous, Complete, Correct, Understandable, Verifiable, Consistent, Traceable and Modifiable. Some of the screenshots are shown in the following.

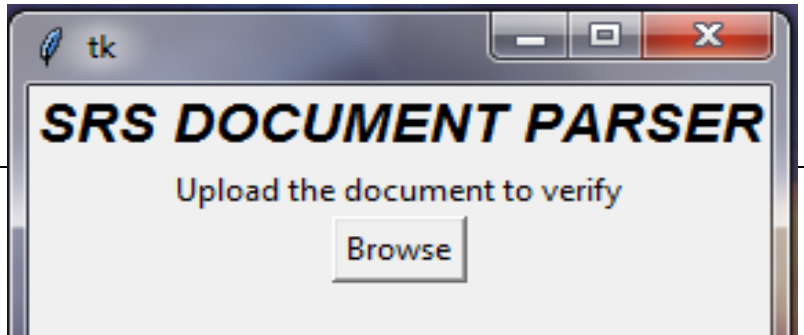

Figure 2 : SRS Document Parser

The User Interface of the SR Document Parser is as shown in figure 2. The screen has options to choose the file (SR doc) and check the options like matched and unmatched entries. The selection the SR document file is depicted in figure 3.

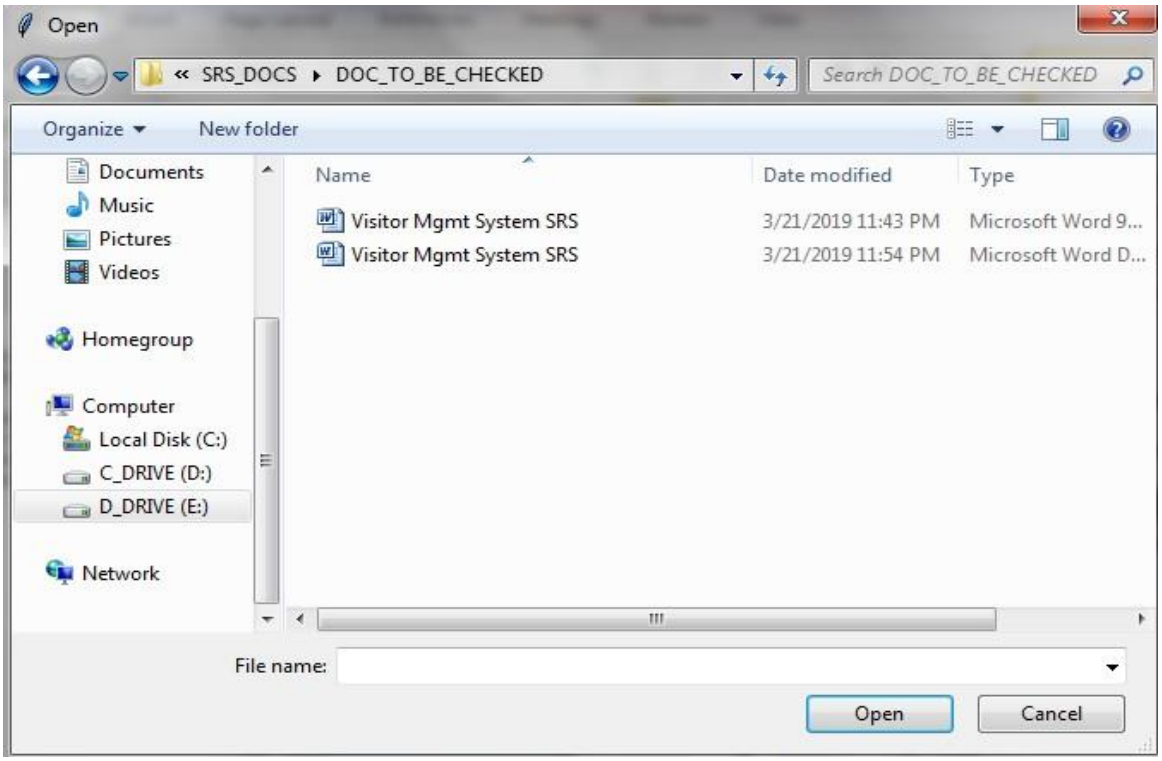

Figure 3 : Selection of SR document

The parsing process searches for the keywords and extracts the matched and unmatched entries. In the proposed work, the document is fed to the developed prototype. The parser in the document checks for the keywords in the document that match with the keywords mentioned in the database. The keywords in the data base are selected keywords which indicate the quality of the document. The keywords available in the data base used for comparison are compared with that of the keywords available in the SR document.

It is observed from the results that following points can be achieved by using the prototype developed.

- Identification of the ambiguities is easier.
- Ambiguous requirements will be efficiently segregated from the other requirements.
- The quality of SR can be ensured and requirements defects will not be propagated to next stages of software development.

6. Conclusion

The current scenario is such that we are largely depended on software in our day-to-day life. Hence many types of software especially the complicated software products are being developed to fulfill the customers. The requirements analyst must make sure that the software requirements are of high quality and the propagation of errors to the next phase of software development is negligible. The requirements analysis stage errors will be propagated to the later stages of the product development which leads to an inefficient and unreliable product. From the literature survey it is clear that Data mining techniques are rarely used for minimizing ambiguities in requirement analysis. Hence, the developed method helps to identify ambiguities in the requirements during the software requirements analysis so that unambiguous and stakeholder agreeable requirements are obtained in minimum time which leads to a better software product.

References

1. Ashima Rani , Gaurav Aggarwal "Algorithm for Automatic Detection of Ambiguities from Software Requirements", IJITEE , ISSN: 2278-3075, Volume-8, Issue-9S, pp. 878-882, July 2019 .
2. Q Riaz , W. H Butt and S Rehman, "Automatic Detection of Ambiguous Software Requirements: An Insight," 2019 5th International Conference on Information Management (ICIM), Cambridge, United Kingdom, pp 1 - 6, 2019.
3. Mohammed Hafeez Osman and Mohammed Firdaus Zaharin, "Ambiguous Software Requirement Specification Detection: An Automated Approach", International Workshop of Requirement Engineering and Testing, ACM , New York. USA, pp 33-40, 2018.
4. Ali Olow Jim'ale Sabriye, Wan Mohd Nazmee Wan Zainon, "An approach for detecting syntax and syntactic ambiguity in software requirement specification", Journal of Theoretical and Applied Information Technology, Vol.96. No 8, pp 2275-2284, 2018.
5. T Hovorushchenko and O Pomorova, "Methodology of evaluating the sufficiency of information on quality in the software requirements specifications", IEEE 9 th International Conference on Dependable Systems , Services and Technologies Kiev, pp 370-374, 2018.
6. Tetsuo. Tamai1 and Taichi. Anzai, "Quality Requirements Analysis with Machine Learning", in the Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering, pp 241-248, 2018.
7. R Navarro-Almanza, R Jurez-Ramrez and G Licea, "Towards Supporting Software Engineering Using Deep Learning: A Case of Software Requirements Classification", 2017 5 th International Conference in Software Engineering Research and Innovation Mérida, pp 116-120, 2017.
8. Azlin Nordin, Nurul Husna Ahmad Zaidi and Noor Asheera Mazlan, "Measuring Software Requirements Specification Quality", Journal of Telecommunication , Electronic and Computer Engineering, Vol 9, No 3- 5, pp 123-128 , 2017.
9. Uzma Noorin, & Mehreen Sirshar, "Quality Assurance in Requirement Engineering", Global Journal, of Computer Science and Technology C Software and Data Engineering , Vol 17 Issue 1 V1.0, pp 1-6, 2017.
10. A Takoshima and M Aoyama, "Assessing the Quality of Software Requirements Specifications for Automotive Software Systems", 2015 Asia-Pacific Software Engineering Conference (APSEC), New Delhi, pp 393-400, 2015.
11. Shalini Ghosh , Daniel Elenius , Wenchao Li, Patrick Lincoln, Natarajan Shankar, Wilfried Steiner, "Automatically Extracting. Requirements Specifications from NaturalLanguage".2014.
12. https://arxiv.org/abs/1403.3142v1
13. Prerana Chaithra, Shantharam Nayak, "Quality Assurance Techniques in SRS Documents", Elsevier-Scopus indexed International Journal of Innovative Technology and Exploring Engineering, ISSN: 2278-3075, Volume-9, Issue-2S, December 2019
14. P Bourque and R E Fairley, "Guide to the Software Engineering Body of Knowledge (SWEBOK (R)): Version 3.0". IEEE Computer Society Press. pp 27- 40, 2014.
15. Prerana Chaithra, Surekha R., Shantharam Nayak "Influence of Mathematics on Software Engineering", International Journal of Management Technology and Engineering, , ISSN: 2249-7455, Vol-9, Issue-5, May 2019, pp.3541-3547
16. Prerana Chaithra, Shantharam Nayak, NandaKumar A. N. " Segmentation and Clustering of Stakeholders using Data Mining Techniques ", Advanced International Journal, ISBN -13915, ISSN: 978-81-936820-0-5, Vol-1, June 2018, pp

482-488

17. Prerana Chaithra, Vijayalakshmi, Shantharam Nayak, "Analysis of Various Methodologies for reducing Ambiguities in Software Analysis", Indian Technology Congress 2014, Nimhans Convention Centre, Bangalore, 21-22 August 2014, pp. 29.
18. Yue Wang, Irene L Manotas Gutierrez, Kristina Winbladh, and Hui Fang, "Automatic Detection of Ambiguous Terminology for Software Requirements", International Conference on Application of Natural Language to Information Systems, pp 25-27, 2013.
19. Bechoo La l, Dr. Chandrahauns R. Chavan, "An Optimization Approach to Analysis of Requirement", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 2, pp 311-316, February 2013.
20. Fatwanto A., "Software requirements specification analysis using natural language processing technique", International Conference on QiR (Quality in Research) at Yogyakarta, IEEE, print ISBN 978-1-4673-5784-5, pp. 105 – 11025-28 June 2013.
21. Misra J., Das S., "Entity Disambiguation in Natural Language Text Requirements", 20th Asia-Pacific Software Engineering Conference, IEEE, Volume 1, pp. 239 – 246, 2013.