IJCSE International Journal of Computer Sciences and Engineering Open Access

Survey Paper Vol.-6, Issue-12, Dec 2018 E-ISSN: 2347-2693

A Systematic Literature Survey for Detecting Ambiguity in SRS Using Artificial Intelligence

Shruti Mishra[1*], Vijay Birchha[2], Bhawna Nigam[3]

[1]Department of Computer Science & Engineering, Swami Vivekananda College of Engineering, Indore, India

[2]Department of Computer Science & Engineering, Swami Vivekananda College of Engineering, Indore, India

[3]Department of Information Technology, IET DAVV, Indore, India

[*]*Corresponding Author: sm.mishra103@gmail.com, Tel.: +917000399236*

Available online at: www.ijcseonline.org

Accepted: 12/Dec/2018, Published: 31/Dec/2018

***Abstract*—**Research in recent years has shown integration amongst the significant and dynamic areas of software engineering and semantic web engineering. The success of any software system is depending on how well it meets the requirements of the stakeholders. A software requirement specification written in natural languages, are basically ambiguous, which makes the documentation unclear. Due to unclear requirements, software developers develop software, which is different from the expected software based on the customer needs. Therefore, well documented requirements should be unambiguous and it is possible only when it has only one meaning.The main purpose of this research is to propose a technique that is able to detect ambiguity in software requirements specification document automatically using artificial intelligence. To validate the outcome of the proposed work, generated result of the proposed work will be evaluated and validated by making the comparison between the proposed prototype results, previous ambiguity detection framework and human-generated results to decide how the proposed work is more efficient and reliable for ambiguity detection.

***Keywords*—**Software Requirements Specification, Artificial Intelligence, Deep Learning, Ambiguity Detection.

I. Introduction

Technology is becoming more important in our daily routine life. Probably the most evident examples are computers (e.g. laptops, smartphones, tablets, smart watches, and many other gadgets) and the internet, which are essential in many (if not every) activities we do. All of these devices need software (programs) that tells these devices what to do, when to do it, how to work, etc. Software development is a multifarious and sophisticated task that involves enormous efforts from numerous participants and produces a large amount of information [1]. By reuse of already available knowledge, can save efforts in the development and maintenance of software systems [2].Also, when software development teams are present at different geographical and virtual locations, it may lead to the generation of inconsistent information due to lack of proper knowledge sharing mechanism. Thus, for large software system development, reusing and sharing software engineering knowledge turns out to be a major operative challenge, which motivates researchers to explore possible supporting technologies[3][4]. Integration among research fields of Semantic Web Engineering & Software Engineering (SE) has been shown by recent studies, illustrating the advantages of collaborating semantic techniques with Software Engineering. A rising trend to utilize ontology to exchange and interconnect Software Engineering knowledge across the Web has been identified by the Software Engineering community [5][6]. "However, this integration still possesses various issues and challenges that ought to be addressed. Issues and challenges, which will keep this integrated field dynamic and lively for years to come" [7][8][9][10]. Among such issues, one is to detect ambiguities in Software Requirements Specification (SRS). SRS is always the first deliverable aspect for any software product. It plays as the "parent" document because all successive project management documents, such as Software Design Document (SDD), software architecture specifications, testing & validation plans, and documentation manual, are related to it [11]. A good SRS has some desirable characteristics, i.e., it is expected to be correct, complete, verifiable, unambiguous, consistent, ranked for importance and stability. Usually, SRS is written in general purpose natural language which is inherently ambiguous. The literal meaning of the word unambiguous is: "not having multiple possible meanings". This implies that each requirement is expected to have one and only one interpretation. One way to get rid of ambiguities is also to use formal language, but it is very complex, time-consuming and expensive [12]. We proposed a technique that uses SRS (written in the natural language) as input and detects possible ambiguity automatically using Artificial Intelligence (deep learning) process.

Deep Learning is one of the hottest trends in Artificial Intelligence approached as Deep Learning approaches produced results superior to the state-of-the-art in problematic areas such as natural language processing (NLP), image processing and simulation. To support the growth of the Deep Learning community, several open source projects appeared providing implementations of the most common Deep Learning algorithms. These projects vary in the algorithms they support and in the quality of their implementations [13].

Natural language processing is much easier and efficient when using deep learning. Therefore, we are using deep learning techniques to create a framework that detects ambiguity in given SRS. This study is focused to answer these research questions:

How well Deep Learning techniques can extract relevant knowledge from software requirements specification written in natural language?

How the Deep Learning deal with unstructured data present in SRS.

Is word embedding vector representations enough robust to capture features to classify the requirements?

Rest of the paper is organized as follows, Section I contains the introduction of software engineering, ambiguity in SRS, and deep learning, Section II contains the current approaches for ambiguity detection on the basis of their types, Section III contain the related research work in the area of SRS and ambiguity detection, Section IV explains ambiguity detection and deep learning methodology with appropriate figure, Section V describes results and discussion, Section VI concludes research work with future directions

II. Current Approach

Ambiguity detection is a popular research area in software engineering there are many proposed works have already been used in detecting and correcting incomplete and inconsistent requirements specification [14]. However, we have not seen any work for detecting ambiguities in SRS using deep learning approach. There are several types of ambiguities that can be present in SRS such as lexical ambiguity, syntactic ambiguity, semantic ambiguity, pragmatic ambiguity, vagueness and generality, and language error [15]. A short description of these ambiguities is illustrated below

A. Lexical ambiguity

Lexical ambiguity occurs due to homonymy and polysemy. “Homonymy occurs when two different words have the same written and phonetic representation, but unrelated meanings and different etymologies, i.e., different histories of development”. Examples of Homonyms are bad or bade, accept or except, brake or break, week or weak, etc. “Polysemy occurs when a word has several related meanings but one etymology” [15]. Example of polysemy can be the word ‘green’, which has several different meanings with a common etymology, such as colour green or not ripened or mature.

B. Syntactic ambiguity:

“Syntactic ambiguity, also called structural ambiguity, occurs when a given sequence of words can be given more than one grammatical structure, and each has a different meaning”[15]. An example of syntactic ambiguity is the following sentence:

“I met Bhatia and Kumar and Roy met me.”

The above sentence can be either read as I met (Bhatia and Kumar) and Roy met me or as I met Bhatia and (Kumar and Roy) met me. Each sentence is leading to a different meaning.

C. Semantic ambiguity:

“Semantic ambiguity occurs when a sentence has more than one way of reading it within its context although it contains no lexical or structural ambiguity” [15]. An example of semantic ambiguity is the following sentence:

“The truck hits the pole while it was moving.”

Though it is obvious from the above sentence that the moving truck hits the standing pole as the phenomena is very close to real-world situation, but if the same logical form of the sentences needed to be learnt by the computer, it is not an easy task to supply a real-world model to a computer system.

D. Pragmatics ambiguity:

“Pragmatics is the study of the relations between language and context” [16]. Pragmatic ambiguity occurs when a sentence leads to numerous implications in the context in which it is articulated. An example of pragmatic ambiguity is the following sentence:

“Is the program running?”

The above sentence can be understood in many ways as if someone asked, Is the program being executed in a computer or Is the program being aired on TV? The implication depends upon in what situation one is asking and to whom he is asking.

E. Vagueness and generality:

Vagueness and generality occur when words or phrases are open to more than one interpretation. A word may have either general or ambiguous meaning. An example of vagueness or generality is the following sentence:

"I am coming to the bank."
The word 'bank' can have more than one interpretation, as in river bank or in a money bank.

F. Language error ambiguity:
"A language error ambiguity occurs when a grammatical, punctuation, word choice, or other mistakes in using the language of discourse leads to text that is interpreted by a receiver as having a meaning other than that intended by the sender" [15]. Here the sender is not aware of the fact that the error has been committed, and the receiver may or may not be aware of the fact that error has been executed. Among all the ambiguities defined above, lexical and syntactic ambiguities have already been detected using NLP techniques [17]. We intend to automate and improve lexical and syntactic ambiguity detection and examine the remaining significant ambiguities for improving requirements that assist in quality software development.

III. RELATED WORK

Many researchers tried to solve the ambiguity problem in NL SRS using different methods and techniques. These techniques can be summarized into three main categories: Unified Modeling Language (UML) based, Ontology-based and Natural Language Processing (NLP) based techniques [18].

UML based technique was used as a method to protect against the ambiguity and its impact in the software development lifecycle. This approach based on developing UML activity diagram and its equivalent model in a window navigation format for mapping initial task to workflow document based on the relationship between them. In addition to that, there is a table called traceability table that contains the candidate requirement names. In this table, the stakeholder view will be built before the activity diagram. In a summary, this method stresses keeping the requirements as the expected outcome. The researchers of this method clearly stated that lack of evaluation can become one of the limitations of their research [19].

Ontology-based is a requirements analysis method using domain ontology that allowed the software engineers to detect ambiguity in SRS document. This method enables to measure the document quality and able to predict the future changes in the SRS documents. The ontology-based technique is a lightweight semantic technique used to detect and solve ambiguity exists in documents written in natural language format. In this method, the ambiguity is detected by mapping requirement to multiple unrelated elements. In this method, Ambiguity is detected by mapping requirements into several elements that are not related [20].

Many researchers tried to solve NL SRS ambiguity using natural language processing. These techniques include a Dowser prototype tool. The dowser prototype tool is designed to identify ambiguities exist in the SRS document. Initially, Dowser tool parses the requirements using constraining grammar. In addition to that, the object-oriented analysis model of the system will be developed by creating its classes, methods, variables and associations. Lastly, the model will be presented for the reviewers to detect the ambiguity. In the literature, there is no much work done in automatically detecting NL SRS ambiguity using parts of speech tagging technique. The following section discusses how the research is conducted.

IV. METHODOLOGY

To detect ambiguity existing in the software requirements documents proposed methodology contains four main phases: initial investigation and analysis, train the system, feature extraction, implementation and conclusion.

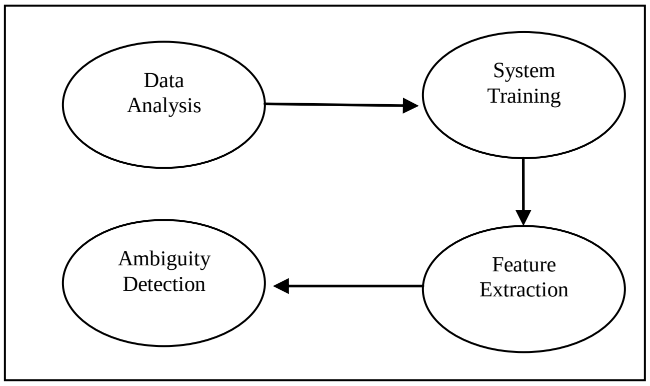

Figure 1. Proposed Ambiguity Detection Life Cycle

First of all, we analyse the NL SRS by using existing literature related to the study has been reviewed to get enough idea about requirements engineering techniques used to elicit the requirements, how the requirements can be documented in natural language format. In this phase, ambiguity problems in NL SRS are explained. Also, related work used to solve ambiguity problems has been discussed. The new proposed framework for automatically ambiguity detection is being presented.

Second, based on the first phase we train the system with collected information and data.

Third, based on training data proposed system generates features related to inputs (NL SRS or Diagrams or Images) and learn from related features and automate self to detect ambiguity in SRS. Implemented system uses these features to match the possible ambiguities and find the appropriate ambiguity.

V. Results and Discussion

The following steps are implemented to detect ambiguity in NL SRS document. These steps classify the ambiguity into syntactic and syntax ambiguities.

1) Retrieve the NL SRS document and related images or diagrams.
2) Create sets of words and phrases, extract data from images if any exists.
3) Train the system with related data.
4) Extract the features from the retrieved SRS.
5) Match the extracted features with the trained system.
6) Detect ambiguity and store in a data structure.
7) Check if any sentence is present in the matching list.
8) Calculate the total number of syntactic and syntax ambiguities.
9) Calculate the percentage of ambiguities detected.

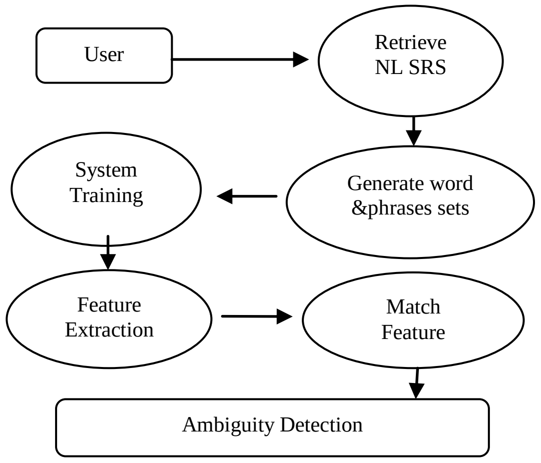

Figure 2: Proposed Architecture of Ambiguity Detection System.

VI. Conclusion and Future Scope

A deep learning based framework, as well as an implementation approach for detecting ambiguities in SRS, is presented in this research paper. This research work will facilitate eliminating significant ambiguities for improving requirements that assist in quality software development. As a potential direction of future work, we intend to compare and evaluate our approach with the original SRS using a case study.

References

[1] R. Beniwal. "Analysis of Testing Metrics for Object Oriented Applications." In Computational Intelligence & Communication Technology (CICT), 2015 IEEE International Conference on, pp. 41-46. IEEE, 2015.
[2] K. Sharma, R. Garg, C. K. Nagpal, and R. K. Garg. "Selection of optimal software reliability growth models using a distance-based approach." Reliability, IEEE Transactions on 59, no. 2, pp. 266-276, 2010.
[3] K. S. Kaswan, S. Choudhary, and K. Sharma. "Software Reliability Modeling using Soft Computing Techniques: Critical Review." J Inform Tech SoftwEng 5, no. 144, 2015.
[4] R. Studer, R. Benjamins, and D. Fensel, "Knowledge engineering: Principles and methods," Data & Knowledge Engineering 25, no.1, pp. 161–198, 1998.
[5] HJ Happel and S Seedorf. "Applications of ontologies in software engineering." In Proc. of Workshop on Sematic Web Enabled Software Engineering"(SWESE) on the ISWC, pp. 5-9. 2006.
[6] Y Zhao, J Dong and T Peng, "Ontology classification for semantic-webbased software engineering,Services Computing, IEEE Transactions on Services Computing", Vol. 2, No. 4, pp. 303-317, 2009.
[7] D Gaševiü, N Kaviani and M Milanoviü. "Ontologies and software engineering." In Handbook on Ontologies, pp. 593-615. Springer Berlin Heidelberg, 2009.
[8] M.P.S Bhatia, A Kumar, and R Beniwal, "Ontologies for Software Engineering: Past, Present, and Future,"pp 232-238 IEEE , 2016.
[9] M.P.S. Bhatia, R. Beniwal and A. Kumar, "An ontology-based framework for automatic detection and updation of requirement specifications." In Contemporary Computing and Informatics (IC3I), 2014 International Conference on, pp. 238-242. IEEE, 2014.
[10] M.P.S. Bhatia, A. Kumar, and R. Beniwal, "Ontology Based Framework for Automatic Software's Documentation." In Computing for Sustainable Global Development, 2015 2nd International Conference on, pp. 725-728. IEEE. 2015.
[11] B S. Dogra, K Kaur, and D Kaushi. Enterprise Information Systems in 21st Century: Opportunities and Challenges. New Delhi: Deep and Deep Publications, 2009.
[12] S Armitage, "Software Requirement Specification." 1996. http://www4.informatik.tu-muenchen.de/proj/va/SRS.pdf (Last accessed date: October, 2015)
[13] Navarro-Almanza, Guillermo Licea "Towards Supporting Software Engineering Using Deep Learning: A Case of Software Requirements Classification" Software Engineering Research and Innovation (CONISOFT), 2017 5th International Conference in, IEEE 2017
[14] Yu Kai, Jia Lei, Chen Yuqiang et al., "Deep Learning: Yesterday Today and Tomorrow[J]", Journal of Computer Research and Development, vol. 50, no. 9, pp. 1799-1804, 2013.
[15] SC. Levinson, "Pragmatics (Cambridge textbooks in linguistics)." 1983.
[16] A Nigam, N Arya, B Nigam and D Jain. "Tool for Automatic Discovery of Ambiguity in Requirements," IEEE 2012.
[17] Sandhu, G. and S. Sikka. State-of-art practices to detect inconsistencies and ambiguities from software requirements. in Computing, Communication & Automation (ICCCA), International Conference in,IEEE 2015.
[18] A. Aamodt, E. Plaza, "Case-Based Reasoning: Foundational Issues Methodological Variations and System Approaches", Artificial Intelligence Comm., vol. 7, no. 1, pp. 39-59, 1994.
[19] Hagal, M.A. and S.F. Alshareef. A systematic approach to generate and clarify consistent requirements. in IT Convergence and Security (ICITCS), International Conference in,IEEE 2013.

[20] D.M. Berry, E. Kamsties and M.M. Krieger "From contract drafting to software specification: Linguistic sources of ambiguity-a handbook version 1.0."(2003). http://cs.uwaterloo.ca/~dberry/handbook/ambiguityHandbook.pdf

Authors Profile

Miss Shruti Mishra pursued Bachelor of Engineering from Shri Dadaji Institute of Technology and Sciences, Khandwa.She is pursuing Master of Engineering in Computer Engineering from Swami Vivekananda College of Engineering, Indore.She has a research interest in Software Engineering; Internet of Things, Computer Networking, Machine Learning and Computer Vision.

*Mr. Vijay Birchha*is working as Assistant Professor in Department of Computer Sciences and Engineering at Swami Vivekananda College of Engineering, Indore.He has completed Bachelor of Engineering from Institute of Technology, Guru Ghasidas University, Bilaspur (C.G) and Master of Engineering in Computer Engineering from Institute of Engineering & Technology, DAVV, Indore. He is currently pursuing Ph.D. in Computer Engineering from Institute of Engineering & Technology, DAVV, Indore. He has various IBM profession certifications. He has teaching experience more than fourteen years. He has research interest in Software Engineering; Cloud Computing, Computer Vision and Big Data Analytics.

Dr. Bhawna Nigam is working at the post of assistant professor in Department of Information Technology, Institute of Engineering & Technology, Devi Ahilya University, Indore, India. She has completed her PhD, Post-graduation and graduation in Computer Engineering from Institute of Engineering & Technology, DAVV, Indore. She has certified by NVIDIA DLI for Image Classification Using DIGITS. She has awarded for bestwomen in research award by RSRI conference and mentor awards by Edureka. She has delivered various expert lectures on Artificial Intelligence and Data Science. She has teaching experiencemore than fourteen years. She has research interest in Data Mining, Machine Learning, Deep Learning and Big Data Analytics.