International Journal of Synergy in Engineering and Technology
IJSET

A Tool to Detect Pragmatic Ambiguity with Possible Interpretations Suggestion in Software Requirement Specifications

Khalid Abdikarim Mohamed, Jamilah Din and Salmi Baharom

Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Selangor

*Corresponding author: jamilahd@upm.edu.my

KEYWORDS

Requirement Engineering
Ambiguity
Detection
Suggestion
NLP
SRS

ABSTRACT

The success of any software must meet the stakeholder's expectations and needs. Requirement specifications are collected from clients and documented in software requirements' specifications. Most software requirement documents are written in natural languages. Natural language requirements may be inconsistent and ambiguous. Such inconsistencies and ambiguities can result in misinterpretations and incorrect implementations in designing and developing of software. The purpose of this study is to detect and correct ambiguity in software requirements. Ambiguity is a statement that has more than one meaning. The most common types of ambiguities are semantic, syntactic, lexical, syntax, and pragmatic. Therefore, this study aims to estimate the degree of ambiguity of typical computer science words (e.g., system, application, database) when used in different domains. Word embedding is used for identifying and detecting ambiguous words in a requirement. Then the study uses linked data for resolving ambiguity words. To evaluate the proposed tool, open-source software specification documents was used to check its performance by comparing human detection and correction capacity and the outcome demonstrates that humans have difficulty detecting and correcting ambiguity in software requirement specifications (SRS) from one domain to other domains as compared to the proposed tool. The result shows that the developed tool was accurate in detecting and correcting pragmatic ambiguities, compare to human.

1.0 Introduction

Software systems are currently important to all society in the world, such as education, manufacturing, government, social networking, sports, banking, and health care. It takes time,

Received October 2022; received in revised form November 2022; accepted December 2022

cost, tools, concepts, infrastructure, and experts to develop a software system. Incorrect requirements can trigger the project's failure [1].

Requirement Engineering becomes the foundation of the stakeholders' needs. Requirement Engineering is the process of describing the requirements of the system by means of predefined activities to elicit users' needs through collecting, analyzing, modeling, validating, documenting, and managing the requirements [2]. It is considered a successful project if the software system has the same functionality and features as defined in the SRS document. SRS explains the software's main content to design and develop it [3].

There are two ways of documenting requirements: documenting using formal approaches and using non-formal approaches. Both approaches have their advantages and disadvantages. Formal approaches are more trustworthy and reliable because they follow specific formal and mathematical notations, which limit the chance of ambiguities. However, they are less communicative, and most customers and stakeholders are unable to understand them. In contrast, informal documentation uses natural language to document the requirements that is less reliable and more vulnerable to ambiguities. Regardless, using informal documentation approaches is preferred over formal documentation [4].

The usage of natural language might lead to ambiguity. Ambiguity is a word or sentence that is not clear about the meaning or has more than one interpretation [5]. The most common types of ambiguities are semantic, syntax, syntactic, lexical, and pragmatic ambiguity [6].

Lexical ambiguity is when a single word has many meanings. Lexical ambiguity occurs when a word has different meaning, for example the word “young” means inexpert or young of age [5]. Syntactic ambiguity arises when the sequence of words (sentence) has multiple parse trees or if there are multiple grammatical structures in the statement [2]. For example the sentence: I saw a girl with a telescope. Two different sentences can be parsed: a) the girl has a telescope with her. b) I used a telescope to see the girl.

Semantic ambiguity arises as soon as the sentence has several understandings without lexical, syntax, and syntactic ambiguity and often happens when the scope of all things involves the scope of one thing. In other word, it occurs when the sentence have more than one meaning based on their context [6]. For example, all users enter a password to access the website. This example can be translated as: a) means all users enter the same password to access the website, b) each user enters unique password to access the website.

Syntax ambiguity is a type of ambiguity arises when a statement does not end with a full stop (“.”) and if the doer is not stated in the sentence [2].

The last type is pragmatic ambiguities, which happen when different readers give different interpretations or meaning to the requirement in the specific context. It depends on the context of the requirement, includes the domain knowledge of the person reading the requirement. It affects the understandability of the requirement, the commonsense knowledge, the viewpoint, and the reader’s background knowledge [7].

Software built for multi -purpose. Software is also used for technical areas, which are specific to a particular skill. Sometimes systems analysts do not understand the technical terms of a domain. Statements are written based on the experience of systems analysts, which may not be appropriate for the domain. This will result in inaccurate software requirements statements. Pragmatic ambiguity is ambiguity associated with a domain. When this happens, the software being built cannot meet the actual needs of the user. This study focuses on natural language ambiguity in SRS, especially pragmatic ambiguity detection and correction in natural language processing. The study estimates the degree of ambiguity of common computer science terms such as application, system, and database when used in different domains. Word embedding technique is used to detect the ambiguous words. Word embedding is a strong approach for analyzing natural language that has been widely utilized in information retrieval and text mining. It can capture the context of a word in a document and measure the relations with other words. The meaning of the word is given by the nearest possible meaning and the company the word keeps

[10]. Similarly, linked data was used for resolving ambiguity words. It presents the meaning of ambiguity word and contains clear synonyms in a range of different languages [8].

2.0 Methodology

The methodology of this study is divides into four phases, which are preliminary investigation and analysis, identified approaches used in this study, tool development, and lastly evaluating the effectiveness of the tool.

Phase1: reviewed the related work of this study, includes natural language (NL) ambiguities and their types and how it can be detected, corrected. The study then narrowed down to pragmatic ambiguity to identify techniques to detect and correct pragmatic ambiguity in SRS. The study then analyzed and reviewed how researcher detect and correct pragmatic ambiguity. Researchers used different approaches and techniques to detect pragmatic ambiguity. These studies indicated that there are five different methods used by researchers to deal with pragmatic ambiguity. These approaches are knowledge base approaches [4], Controlled Language and Rule Based Approach [9], Machine Learning & Heuristics Based Approach [9], Word embeddings-based approaches [1] [10] [11], and shortest-path search algorithm approach [12] [13]. The literatures also indicate that there are three approaches to deal with resolving pragmatic ambiguity: linked data [8], Score-Based Automatic Resolution approach [14], and web data and semantic knowledge approach [15]. The following table 1 shows the comparison between some of exiting tool dealing with pragmatic ambiguity detection and correction and the proposed tool.

Table 1: Techniques to Detect Pragmatic Ambiguity

Feature support	Alessio Ferrari & Esuli, 2019 [10]	Mishra & Sharma, 2019 [1]	Alessio Ferrari et al., 2014 [12]	Our study
Detection approach used	Word Embedding	Word embedding	Shortest-path search algorithm	Word embedding & linked data
Supports Detection	Yes	Yes	Yes	Yes
Supports Correction	No	No	No	Yes
Domain focus	Computer Science, Mechanical Engineering, Electronic Engineering, Sports, and Medicine.	Biomedical engineering, petroleum, environmental, civil engineering, and chemical engineering	graph-based modelling of the background knowledge	Mechanical Engineering (MCEE), Civil engineering (CIVE), Biomedical engineering (BIEE)
User interaction	High	Medium	High	Low

Phase 2 involves design of a new tool using word embedding and linked data approaches to detect and resolves one form of ambiguities, which is pragmatic ambiguity in SRS. Word embedding is use to detect ambiguous words in a requirement, because using word embedding base technique is very useful in detecting context-dependent ambiguities. It can capture a word's meaning and measuring its semantic relationship of similarity with other words in a predefined vector space, and it reflects specific words as actual valued vectors [1]. Words is map to one vector

and depending on the use of words, the vector values are learned. Words that are used in similar ways are likely to have similar meanings. The linked data was also used for resolving ambiguity. It employs databases that stored some distinct word senses to determine context knowledge of requirements. The normalized requirements are taken as input then each word in that requirement will be used to find the ambiguous words (keywords) with the help of database. If there are multiple possible interpretations for the word, that term will be listed with the possible interpretations [8].

At phase 3 a prototype is developed. jQuery was used as Front-End and express Node.js web application framework as Back-End. User can upload documents in .txt and .doc format then the prototype detects ambiguous words from common computer science terms. User can hover the mouse to the detected word and can see the possible interpretations. User can select the correct replacement word. The detected ambiguity and counts of total number of detected and corrected ambiguities can then be saved as document.

The proposed tool is evaluated in phase 4. SRS document is collected from open-source website to evaluate the effectiveness of the new developed tool. Comparison between the developed tool result and human ability to manually detect and correct ambiguity has been carried out.

Figure 1 shows the high-level view of the research method.

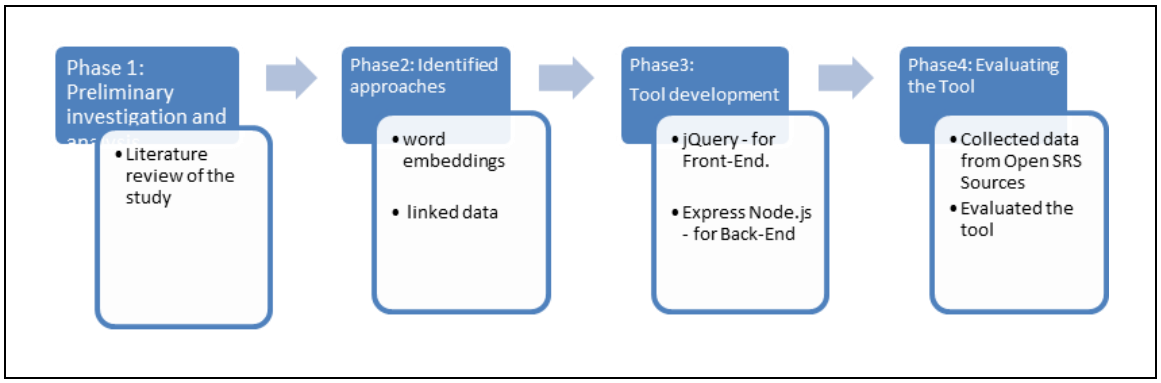

Figure 1: Overview of the Methodology

3.0 Result and Discussion

In order to examine the approach used in this study, a repository of 10 real world requirements for each domain (3) has been retrieved from scribe.com as dataset for evaluation. These requirements consist of 10 requirements from Mechanical Engineering (MCEE), 10 requirements from Civil engineering (CIVE), and 10 requirements from Biomedical engineering (BIEE) domain. Each selected contains pragmatic ambiguity.

These sentences were not created specifically for the purpose of evaluating the tool, rather they were randomly selected from real software requirements in SRS. The following Tables 2, 3, and 4 show the list of referenced requirements from three domains. These requirements have taken from SRS documents stored in scribd.com website as randomly. Scribd.com is an online, open-source website that stores a large number of SRS documents developed in various domains. Several researchers include [16] are used this website as source.

Table 2: The List of Referenced Requirements from Biomedical Engineering (BIEE) Domain

The Outbreak Management system (OM) must be able to electronically record and store data which can be uploaded to the system from remote device
The OM's system must provide the ability to accept digital data.
Systems supporting OM must provide the ability to control the configuration of specific questionnaires
New users should be able to register to the system
administrators shall be able to view and modify all information
The application has a simple GUI and easy to use
The system shall use the MySQL Database, which is open source and free
Any modification (insert, delete, update) for the Database shall be synchronized and done only by the administrator
The system shall be a Web-based application
The system generates reports on the following information

Table 3: The List of Referenced Requirements from Civil Engineering (CIVE) Domain

New users should be able to register to the system
The software is designed for the smart phone web, and standalone desktop PC
The application has a simple GUI and easy to use
Any modification (insert, delete, update) for the Database shall be synchronized and done only by the administrator
The Development environment shall be Windows 2000
The system shall be a Web-based application
The system generates reports on the following information
administrators shall be able to view and modify all information
The system shall keep a log of all the errors
Communication functions require the internet protocol version 6, and it will follow HTTPS.

Table 4: The List of Referenced Requirements from Mechanical Engineering (MCEE) Domain

The application has a simple GUI and easy to use
The system shall use the MySQL Database, which is open source and free
Any modification (insert, delete, update) for the Database shall be synchronized and done only by the administrator
The system shall be a Web-based application
The system generates reports on the following information
The system shall provide the capability to back-up the Data
The systems must conform to the Microsoft Accessibility guidelines
The system shall keep a log of all the errors
Credentials of the user are encrypted through algorithm and application is accessible only to authenticated users
administrators shall be able to view and modify all information

Two evaluation have been carried out as part of this study: using our pragmatic ambiguity prototype tool and manually detect and correct the ambiguity. Both evaluations' results have been analyzed.

The prototype has been developed to detect ambiguity in 3 domains which are: medical, civil engineering and manufacturing engineering. For the purpose of evaluation, ten (10) people been selected. As shown in Figure 2, three (3) of them are from medical domain and five (5) of them

are civil engineering domain. Whereas two (2) people are mechanical and manufacturing engineering. We need three different domain of participants to test the prototype. However we have difficulty to get the same number of participants for each domain. These domain have been chosen in order to examine different context where ambiguities may occur and to detect and correct the ambiguity in the SRS from one domain to other domains.

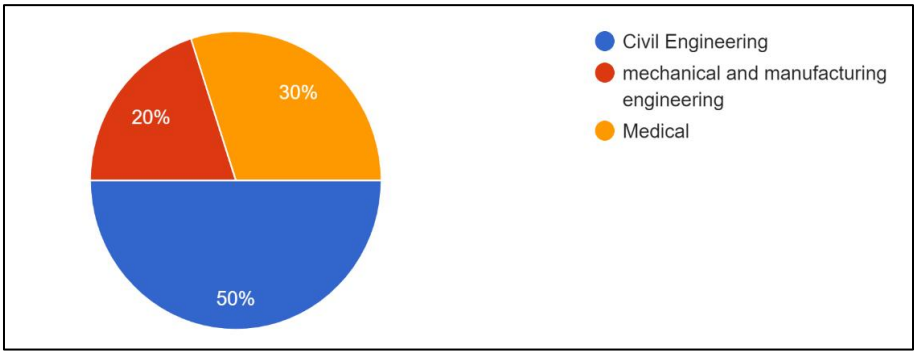

Figure 2: Participants Domain Background

The effectiveness of the developed pragmatic ambiguity prototype tool was evaluated to check it effectiveness. The evaluation's result shows that the tool detects the ambiguities in uploaded SRS documents by highlighting the ambiguity words from computer science terms which may cause different meaning in other domains. The possible ambiguous word is highlighted to red. When user hovering the mouse over the highlighted text, user can see the possible interpretations and suggestion to correct the ambiguities. Furthermore, the tool displays the total number of detected and corrected ambiguities and saves the document. It also shows that the developed tool was accurate for detecting and correcting pragmatic ambiguities in the referenced dataset. The following Figure 3 shows the result generated by the tool.

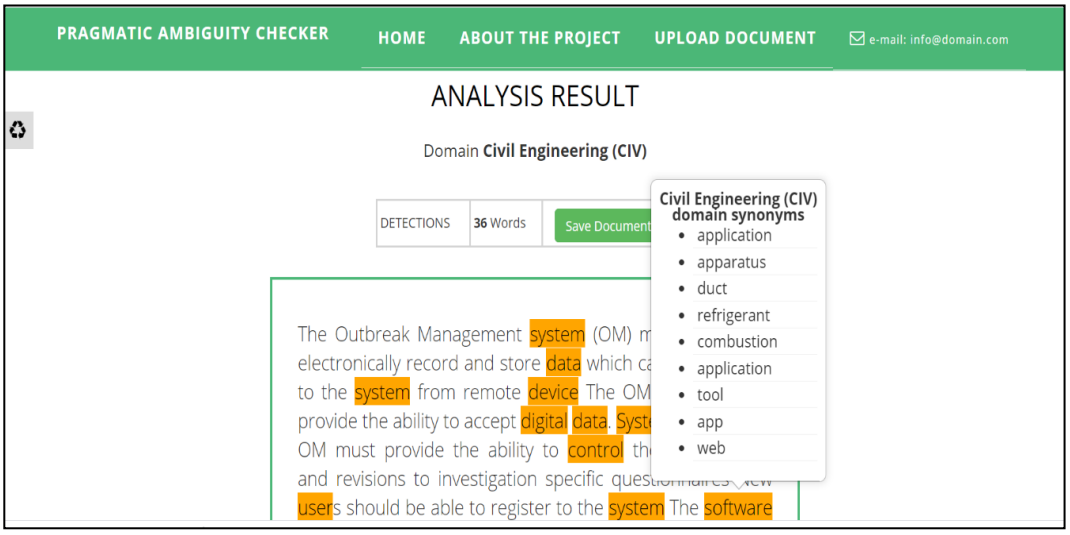

Figure 3: Result Generated by the Tool for Detecting and Correcting Ambiguity

The manual evaluation of detecting and correcting ambiguity in software requirements specifications was done to examine how human can detect the common computer science terms

which have deferent the meaning in other domains and propose possible interpretations to correct the ambiguity.

Participants have been given 10 requirements for each domain that includes computer science terms, which may cause differ the meaning in other domains. These requirements were listed randomly. Each participant expected to choose their domain and check if requirements are none-ambiguous or ambiguous base on their domain. If they found ambiguous terms, they need to give the possible suggestions to resolve the ambiguity.

Each participant selected their domain when started using the tool. Then the requirements related to that domain was displayed. So the participant will check the requirement relevant to their expertise. Result of the evaluation is shown in Figure 4, 5 and 6 for respective domain.

Figure 4 shows for civil engineering (CIVE) domain, 52% of the participants chose the computer science terms in the requirements are ambiguous and differ the meaning whereas 48% chose the requirements are non-ambiguous.

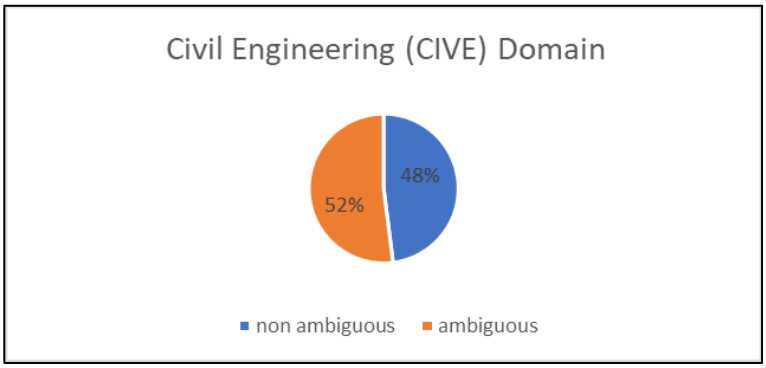

Figure 4: Civil Engineering Domain

Mechanical Engineering (MCEE) domain: The participants need to identify the computer science terminology in 10 requirements and check if the meaning is same or differ in mechanical engineering domain. They then were requested to give the possible suggestions for the ambiguous word. Figure 5 shows that 60% of the participants chose the computer science terms in the requirements are ambiguous and differ the meaning whereas 40% chose the requirements are non-ambiguous.

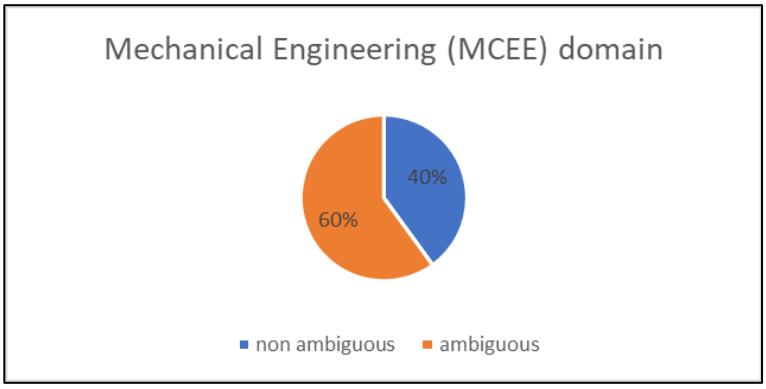

Figure 5: Mechanical Engineering Domain

Biomedical engineering (BIEE) domain: The participants need to identify the computer science terminology in 10 requirements and check if the meaning is same or differ in biomedical

engineering domain. They then were requested to give the possible suggestions for the ambiguous word. Figure 6 shows that 40% of the participants chose the computer science terms in the requirements are ambiguous and differ the meaning whereas 60% chose the requirements are non-ambiguous and the computer science terms not differ the meaning.

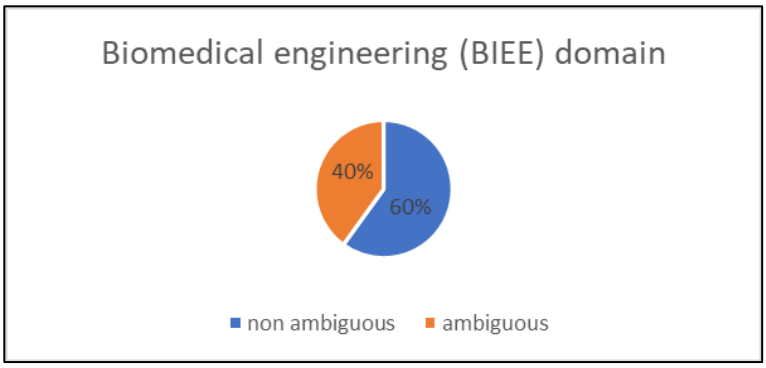

Figure 6: Biomedical Engineering Domain

4.0 Conclusion

Software systems are important to all society in the world, such as education, manufacturing, government, social networking, sports, banking, and health care. It takes time, cost, tools, concepts, infrastructure, and experts to develop a software system. Incorrect requirements can trigger the project's failure. Ambiguous requirement statements is one of the main causes of project failure. The ambiguous requirements must be corrected, or it can result in misinterpretations and incorrect implementations in designing and developing of software.

This project focused on natural language ambiguity in SRS, especially pragmatic ambiguity detection and correction in natural language processing. The prototype tool has boundaries and can be improved in the future. This first version determines the degree of ambiguity of common computer science words when used in three different domains: mechanical engineering (MCEE), civil engineering (CIVE), and biomedical engineering (BIEE). Thus, other domains such as, education, manufacturing, government, social networking, sports, and banking can be added in the extension version. Secondly, the pragmatic ambiguity prototype tool currently accepts in .txt and .doc format. In the future, the tool could be improved to accept more formats, such as PDF. Finally, the tool can only detect one form of ambiguity only and do not consider other types of ambiguity such as: semantic, syntactic, lexical, and syntax. The other types of ambiguities can be added to the tool in the future.

References

[1] Mishra, S., & Sharma, A. (2019). On the use of word embeddings for identifying domain specific ambiguities in requirements. Paper presented at the 2019 IEEE 27th International Requirements Engineering Conference Workshops (REW).

[2] Sabriye, A. O. J. a., & Zainon, W. M. N. W. (2017). A framework for detecting ambiguity in software requirement specification. Paper presented at the 2017 8th International Conference on Information Technology (ICIT).

[3] Ali, S. W., Ahmed, Q. A., & Shafi, I. (2018). Process to enhance the quality of software requirement specification document. Paper presented at the 2018 International Conference on Engineering and Emerging Technologies (ICEET).

[4] Riaz, M. Q., Butt, W. H., & Rehman, S. (2019). Automatic detection of ambiguous software requirements: An insight. Paper presented at the 2019 5th International Conference on Information Management (ICIM).

[5] Abdirashid, A., & Hassan, S. a. (2019). A Tool for Detecting Ambiguity In Software Requirement Specification, International Journal of Advanced Science and Technology, 28.

[6] Ashima, R., & Gaurav, A. (2019). Algorithm for Automatic Detection of Ambiguities from Software Requirements. International Journal of Innovative Technology and Exploring Engineering (IJITEE), 8(9S).

[7] Ferrari, A., & Gnesi, S. (2012). Using collective intelligence to detect pragmatic ambiguities. Paper presented at the 2012 20th IEEE International Requirements Engineering Conference (RE).

[8] Zait, F., & Zarour, N. (2018). Addressing lexical and semantic ambiguity in natural language requirements. Paper presented at the 2018 Fifth International Symposium on Innovation in Information and Communication Technology (ISIICT).

[9] Oo, K. H., Nordin, A., Ismail, A. R., & Sulaiman, S. (2018). An Analysis of Ambiguity Detection Techniques for Software Requirements Specification (SRS). International Journal of Engineering & Technology, 7(2.29), 501-505.

[10] Ferrari, A., & Esuli, A. (2019). An NLP approach for cross-domain ambiguity detection in requirements engineering. Automated Software Engineering, 26(3), 559-598.

[11] Ferrari, A., Donati, B., & Gnesi, S. (2017). Detecting domain-specific ambiguities: an NLP approach based on Wikipedia crawling and word embeddings. Paper presented at the 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)

[12] Ferrari, A., Lipari, G., Gnesi, S., & Spagnolo, G. O. (2014). Pragmatic ambiguity detection in natural language requirements. Paper presented at the 2014 IEEE 1st International Workshop on Artificial Intelligence for Requirements Engineering (AIRE).

[13] Ferrari, A., & Gnesi, S. (2012, 24-28 Sept. 2012). Using collective intelligence to detect pragmatic ambiguities. Paper presented at the 2012 20th IEEE International Requirements Engineering Conference (RE).

[14] Osama, M., Zaki-Ismail, A., Abdelrazek, M., Grundy, J., & Ibrahim, A. (2020). Score-based automatic detection and resolution of syntactic ambiguity in natural language requirements. Paper presented at the 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME).

[15] Khezri, R. (2017). Automated Detection of Syntactic Ambiguity Using Shallow Parsing and Web Data. Master thesis, University od Gothenburg.

[16] Gleich, B., Creighton, O., & Kof, L. (2010, June). Ambiguity detection: Towards a tool explaining ambiguity sources. In International Working Conference on Requirements Engineering: Foundation for Software Quality (pp. 218-232). Springer, Berlin, Heidelberg.