Comparative Analysis of Ambiguities Resolving Tools in Natural Language Software Requirements

Ashima Gambhir
Assistant Professor
Department of Computer Science
Amity University Haryana,India

Abstract-- Requirement Engineering is the primarily important behavior in Software Development Life Cycle(SDLC). In many software systems development projects, the documents available for software requirement analysis are in natural language. Normally, the users express their requirements in natural language statements that primarily come out easy to state. In any case, being expressed in regular language, the announcement of prerequisites frequently will in general experience the ill effects of misinterpretations and off base deductions. Subsequently, the necessities indicated in this manner, may prompt ambiguities in the product details.There are so many techniques and tools are available to determine the ambiguities from Software Requirement documents. This paper presents a state-of-the-art survey and talk about presently available tools for resolving of ambiguity. The main objective of this paper is distributed, divide and examine the research work available in the area, determine matrices for a relative evaluation. Ongoing work, some observations are explained to improving the quality of the requirement analysis process.

Keywords— Requirement Engineering, Ambiguity, Natural language Processing.

I. INTRODUCTION

Requirements Engineering (RE) is the action that includes the capacities related with the extraction, modeling, analysis, verification and specification of the clients necessities [1]. The RE movement frequently begins with the dubiously characterized necessities [2] and results in the end in to a Software Requirements Specification (SRS) record. The industrial software specifier writes the SRS in natural language. Even if a final SRS is written in a formal language, its first draft is usually written in a Natural language(NL). A NL SRS enhances the communication between all the stakeholders. However, on the downside, often a NL SRS is imprecise, unmanaged, indeterminate, inaccurate, unremarkable and ambiguous may ultimately leads to time and cost. An ambiguity can be of different kinds i.e. lexical, semantic, syntactic, pragmatic, vagueness, generality and language error ambiguity.

Manually resolving ambiguity from NL Software Requirements is a time consuming, tedious, expensive and error prone process. An automated and semi automated approach is desirable to resolve ambiguities from software requirement document. There are a number of different tools viz. WSD [11], QuaARS[12], ARM [13], RESI [14], SREE [15, 16], NAI [17, 18], SR-Elicitor [19], and NL2OCL [20] developed to detect and resolve ambiguities. Subsequently, in this paper, we endeavor to display a knowledge into how current ambiguities settling instruments work, the methodology pursued by each apparatus, the kinds of ambiguities settled and the highlights they support. We utilize the presentation estimates, for example, Recall, Precision and F-measure to relatively examine the exhibition of the vagueness settling tools.

II. AMBIGUITY

Ambiguity is termed as competence of being implicit in possible more than two different minds. An vagueness has two sources: missing data and correspondence blunders. Mistakes are sorted in two different ways. The first is creator autonomous blunders – ones, that can be settled without area information for example "syntactic mistakes." The other is creator subordinate blunders – ones that need area information to determine for example "absence of detail" to address the blunder [6, 9].

There are two main categorize of ambiguities i.e. linguistic ambiguities and software engineering ambiguities. Table 1 shows the two main types of ambiguities with subtypes and examples[25].

Table 1: Ambiguity Types

	Ambiguity Types	Subtype
Language Ambiguity	Lexical Ambiguity	Homonym Ambiguity Polysemy Ambiguity
	Syntactic Ambiguity	Analytical_Ambiguity, Attachment_Ambiguity, Coordination Ambiguity, Elliptical Ambiguity
	Semantic Ambiguity	Scope Ambiguity
	Pragmatic Ambiguity	Referential_Ambiguity, Deictic Ambiguity
	Requirements Document Ambiguity	

RE-Specific Ambiguity	Application Domain Ambiguity
	System Domain Ambiguity
	Development Domain Ambiguity
	Vagueness, Language error, Conceptual Translational Ambiguity

A. Lexical Ambiguity: Lexical ambiguity takes place if a word has multiple connotations. Example: "Malika walked to the bank" This could mean that Malika walked to the edge of the river or financial institution.
B. Syntactic Ambiguity: A sequence of words with numerous suitable grammatical interpretations regardless of context . Example: "Quickly read and discuss the tutorial".
C. Semantic Ambiguity: A sentence with more than one explanation in its provided context. Example: Melissa and Freddy are married.
D. Vagueness Ambiguity: A statement that admits borderline cases or relative interpretation. Example: "Freddy is tall".
E. Incompleteness Ambiguity: A linguistically right sentence that gives too little detail to pass on a specific or required significance. Model: "Consolidate flour, eggs, and salt to make new pasta." overlooks some essential data, for example, amount of materials and strategies to be utilized.
F. Referential Ambiguity: A grammatically correct sentence with a reference that confuses the reader based on the context. Example: "The boy told his father about the damage. He was very upset".

III. APPROCHES TO DETECT AND RESOLVE AMBIGUITY

Software Requirements are specified in natural language tend to be ambiguous. Firstly pre-processed the specified document to reduce ambiguity by using Natural language Processing Technique. The possible usage of NLP techniques are: extract requirements from the document, tag the requirements sentence, find duplicate requirements, do the machine translation and extract the ambiguous requirements.

The basic NLP issues are Part-of-Speech (POS) tagging, parsing and ambiguity. POS tagging marks every word of a sentence with predefined parts-of-speech (noun, verb, adjective, etc.).The process of tagging becomes difficult when the word is ambigious. For example:

(a) I want a book.
(b) I want to book a ticket.

In the first sentence (a) the word "book" is a noun and in second sentence (b) the word "book" is a verb. Following are some approaches available to resolve the tag ambiguity.

A. **Rule Based approach**: It is extremely laborious because it requires keeping the rules up to date that cover all cases.
B. **Statistical Based Approach:** It is based on training data set.
C. **Hybrid Based Approach**: It combines the features of both statistical approach and rule based approach.

The author suggest following steps to resolve the ambiguities.

1. Input English Sentences and its UML class model.
2. Parse the sentence using Stanford parser.
3. Perform syntactic analysis.

Figure 1 explains the more different approaches to detect the different types of ambiguities.

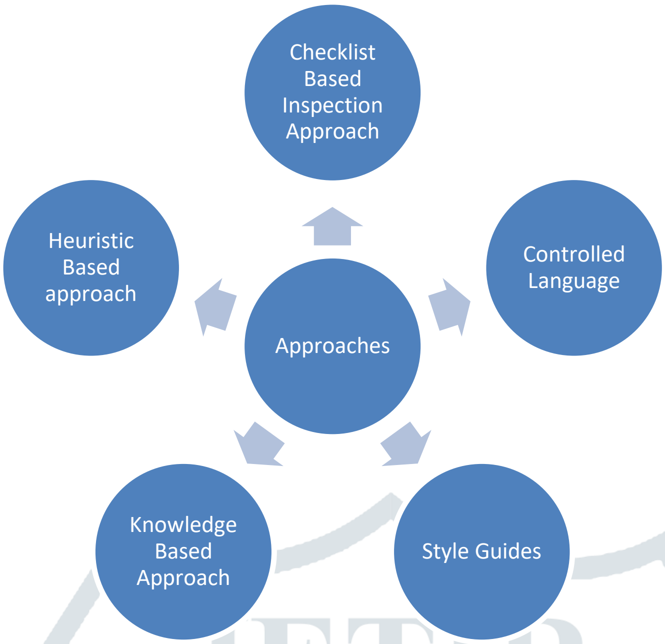

Figure 1:Different approaches to detect ambiguities

IV. COMPARATIVE ANALYSIS OF AMBIGUITY RESOLVING TOOLS

TABLE 2: COMPARATIVE ANALYSIS OF AMBIGUITY RESOLVING TOOLS

Feature Support	Approach	Technology Used	Pre Processing	Concerned Ambiguity	User Interaction	Elicit OOT	Remarks
OOV of NLRS(Automatic)	Knowledge based to ontology	GATE tool , Brill tagger	Yes	Pronoun anaphora	average	True	Non functional requirements elicitation. Ontology generation.
RA in RS via OOM(Semi-Automatic)	Controlled language	Dowser Parser	No	Semantic	average	True	Cannot agreement with modal verbs and negations. Evoke 78.8% (Compound noun) Evoke 93.9%(Single noun)
SREE(Semi-Automatic)	Rule based, Style guide	WordNet, POS tagger	No	Identify Plural, Coordination, Pronoun, Quantifier, Vague	small	False	Report Summary of Ambiguous and incomplete requirements

							statements. Evoke 100%
RESI(Semi-Automatic)	Knowledge Based to ontology	Stanford parser, Cyc, ConceptNet, WordNet	Yes	Avoid Lexical, Scope, Language Error	elevated	True	Input should be in the graph GrGen format.
NAI(Automatic)	Machine learning/heuristics based	LogitBoost, Named entity recognition	Yes	Noun and Verb compound coordination, Anaphora ambiguity	average	False	Establish the Degree of nocuity that the system should tolerate. Accuracy 70% and Evoke 100% (Coordination) Accuracy 82.4% and Evoke 74.2% (Anaphora)
SR-Elicitor(Automatic)	Controlled Language	SBVR, POS Tagger	No	Lexical, Syntactic, Scope-Quantifier	small	True	SBVR rule generation. Recall 80.12% and Precision 85.76%
NL2OCL(Automatic)	Knowledge Based to ontology	SBVR, Stanford parser	No	Attachment, Homonymy	Small	True	A UML class model is required as an input. Evoke 92.85% Accuracy 92.85% (Attachment) Accuracy 99.0% (Homonymy)
CKCO(Automatic)	Knowledge Based to ontology	WordNet, WSD	No	Lexical – Polysemy (ambiguity of nouns)	small	False	Resolved Ambiguity posed to Question Answering (QA) system Precision 83.4%

V. CONCLUSION

Encouraged by the significance of resolving ambiguities, in this paper, I overview the cutting edge in methodologies for settling ambiguities in normal language prerequisites. I survey and talk about various kinds of methodologies and instruments accessible for settling ambiguities in the normal language programming prerequisites determination. I see that the devices are extensively delegated mechanized and semi-computerized. The coherence and comprehend capacity of the necessities increments by applying space explicit language, limited punctuation/syntax and sentence designs.

However, it requires lot of human process, significant expertise and complex to apply in every environment.To identifying the semantic ambiguities, tools that use machine learning approaches and knowledge based to ontology are efficient and give accurate results. However, majority tools use natural language processing tools viz. Stanford parser, dowser parser that is still under improvements. Accuracy of the disambiguation tools depends on the parser they use. If the parser does not recognize the right tokens and their dependencies, then the whole process becomes insufficient, eventually leads to a waste of efforts.

REFERENCES

[1] Sommerville, I. and Sawyer, P. 1997. Requirements Engineering A good practice guide. Chichester: John Wiley & Sons Ltd.

[2] Nuseibeh, B., & Easterbrook, S. 2000, May. Requirements engineering: a roadmap. In Proceedings of the Conference on the Future of Software Engineering (pp. 35-46).

[3] Belev, G. C. 1989, January. Guidelines for specification development. InReliability and Maintainability Symposium, 1989. Proceedings., Annual (pp. 15-21). IEEE.

[4] Christel, M. G., & Kang, K. C. 1992. Issues in requirements elicitation (No. CMU/SEI-92-TR-12). CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST.

[5] Donald G. Firesmith. 2007. Common Requirements Problems, Their Negative Consequences, and Industry Best Practices to Help Solve Them. In Journal of Object Technology, vol. 6, no. 1, January-February 2007, pp. 17-33

[6] Berry, D. M., Kamsties, E., & Krieger, M. M. 2003. From contract drafting to software specification: Linguistic sources of ambiguity. Technical Report, University of Waterloo, Waterloo, Canada. Available at. http://se.uwaterloo.ca/~dberry/handbook/ ambiguityHandbook.pdf

[7] Hutchison, D., Araki, K., Kanade, T., Kittler, J., Kleinberg, J. M., Liu, S. and Weikum, G. 2008. Formal Methods and Software Engineering: 10th International Conference on Formal Engineering Methods, ICFEM 2008, Kitakyushu-City, Japan, October 27-31, 2008. Proceedings. Berlin, Heidelberg: Springer Berlin Heidelberg.

[8] Wordsworth, John B. 1992. Software development with Z: a practical approach to formal methods in software engineering. Addison-Wesley Longman Publishing Co., Inc., 1992.

[9] Kamsties, E., & Peach, B. 2000, December. Taming ambiguity in natural language requirements. In Proceedings of the Thirteenth International Conference on Software and Systems Engineering and Applications.

[10] Zave, Pamela. 1997. Classification of research efforts in requirements engineering. ACM Computing Surveys (CSUR) 29.4 (1997): 315-321.

[11] Nancy Ide and Jean Véronis. 1998. Introduction to the special issue on word sense disambiguation: The state of the art. Computational Linguistics - Special issue on word sense disambiguation, Volume 24 Issue 1,2-40.

[12] Fabbrini, F., M. Fusani, S. Gnesi, and G. Lami. 2001. The Linguistic Approach to the Natural Language Requirements Quality: Benefit of the use of an Automatic Tool. SEW'01 proceeding of the 26th annual NASA Goddard Software En gineering Workshop, IEEE Computer Society Washington, DC, USA, 97.

[13] Willis, Alistair, Francis Chantree, and Anne De Roeck. 2008. Automatic Identification of Nocuous Ambiguity. Research on Language &Computation, (3-4), 1-23.

[14] Sven Körner and TorbenBrumm. 2009. RESI-A natural language specification improver. IEEE International Conference on Semantic Computing (ICSC).

[15] Sri Fatimah Tjong. 2008. Avoiding ambiguity in requirements specifications.Thesis submitted to the University of Nottingham for the degree of Doctor of Philosophy.

[16] Tjong, Sri Fatimah, and Daniel M. Berry. 2013. The Design of SREE—A Prototype Potential Ambiguity Finder for Requirements Specifications and Lessons Learned. Requirements Engineering: Foundation for Software Quality. Springer Berlin Heidelberg, 2013. 80-95.

[17] Hui Yang, Alistair Willis, Anne De Roeck, Bashar Nuseibeh. 2010. Automatic Detection of Nocuous Coordination Ambiguities in Natural Language Requirements. Proceedings of the IEEE/ACM
international conference on Automated software engineering, 5362.ISBN: 978-1-4503-0116-9. DOI=10.1145/1858996.1859007.

[18] Hui Yang, Anne de Roeck ,Vincenzo Gervasi, Alistair Willis Bashar Nuseibeh. 2011. Analyzing anaphoric ambiguity in natural language requirements. Requirements Engineering - Special Issue on Best Papers of RE'10: Requirements Engineering in a Multifaceted World, Volume 16 Issue 3, 163189.DOI=10.1007/s00766-011-0119-y.

[19] Basili, Victor R., Scott Green, Oliver Laitenberger, Filippo Lanubile, Forrest Shull, Sivert Sorumgard. 1995. The Empirical Investigation of Perspective-Based Reading. Technical report the empirical investigation of perspective based reading.

[20] Imran SarwarBajwa. 2012. Resolving Syntactic Ambiguities in Natural Language Specification of Constraints. CICLing'12 Proceedings of the 13th international conference on Computational Linguistics and Intelligent Text Processing, Volume 1, 178-187.

[21] Abderrahman Matoussi and RégineLaleau. 2008. A Survey of Non-Functional Requirements. In Software Development Process Technical report TR-LACL-2008-7, LACL (Laboratory of Algorithms, Complexity and Logic), University of Paris-Est (Paris 12).

[22] Heitmeyer, Constance L., Ralph D. Jeffords, and Bruce G. Labaw., 1996. Automated Consistency Checking of Requirements Specifications. ACM Transactions on Software Engineering and Methodology (TOSEM), ACM vol. 5, no. 3, 231–261. DOI=>10.1145/234426.234431.

[23] Ryan and K. 1993. The role of natural language in requirements engineering. Proceedings of the IEEE Int. Symposium onRequirements Engineering. San Diego, CA, 240-242.

[24] Kof and L. 2004. Natural Language Processing for Requirement Engineering: Applicability to large Requirements Documents. Available at. http://www.dsl.uow.edu.au/~jp989/Scalability_ WITSE04.pdf [25] Eric Brill. 1995. Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging. Journal computational linguistics volume 21 issue 4 December, 543-565.

[25] Ashima rani and Gaurav Aggarwal 2018. Advanced Practices to detect ambiguities and inconsistencies from software requirements. Proceedings of the IEEE Int. Conference on system modelling & advancement in research trends,17-21.