International Journal of Intelligent Computing and Information Science

DETECTING AND RESOLVING AMBIGUITY APPROACH IN REQUIREMENT SPECIFICATION: IMPLEMENTATION, RESULTS AND EVALUATION

Somaia Osama Mostafa Aref

Computer Science Department, Faculty of Computer and Information Science, Ain Shams University
Cairo, Egypt
somaia.osama.r@gmail.com aref_99@yahoo.com

Abstract: *Requirements documentsare always written in natural language. At the point when a sentence can be understood diversely among various readersambiguity is happened [1]. In this paper, we illustrate an automated tool for detectingand resolvingambiguities thatcause a high risk of misunderstanding byseveralreaders and lead to confusion, waste of both effort and time and rework. Sentences in a natural language requirements specification document thathaveambiguity are initialdetected automatically from the text andambiguity type is determined. Sentences thatincludeambiguity are thenresolved automatically also by resolving algorithm based on a set of rules that we collected from training data. We implemented a tool for Detecting and Resolving Ambiguity (DARA), in order to clarifyand estimate our approach. The tool focuses on Lexical, Referential, Coordination, Scope and Vague ambiguity.We determine on the results of a collection of requirement specification documents to evaluatethe performance and utility of the approach.*

Keywords: *Ambiguity, ambiguity detection, ambiguity resolving, disambiguation, natural language processing (NLP), requirement engineering, software requirement specification,*

1. Introduction

The Software Requirements Specification (SRS) is a part of the contract and it must define the user and the system requirements clearly, precisely and unambiguously [2]. The SRS that has inconspicuous, incomplete, unmanaged, unspecified, inaccurate or ambiguous requirement definition may eventually lead to cost and time overruns [3, 4, and 5]. Ambiguity is the possibility to understand a phrase or word in different ways. It is one of the issues that happen in natural language documents. An ambiguity has two sources: communication faultsand inadequate information. Some errors can be resolved without domain knowledge like grammatical error, though some errors need domain knowledge like the lack of details that the user needs. The Ambiguity Handbook [6] presents different types of ambiguities, categorized as Lexical, Syntactic, Semantic, Pragmatic, Vagueness, Generality and Language Error. Although the fact that the

requirements specified in natural language tend to inappropriate interpretations, the requirements are most often specified in natural language[7, 8]. So, it is necessary to develop the approaches whichhandle the ambiguities in user requirement specifications. Manually detecting and resolving ambiguity from software requirements is a boring, time consuming, cause errors, and therefore expensive process. So, an approach to detect and resolve ambiguities automaticallyfrom the requirements statement is needed.

2. DARA Architecture

This section provides an architectural description of the DARA system. It was developed to be modular, extensible, and simple to utilize. We develop an automated system to detect and resolve ambiguities from full text documents. The DARA architecture is shown in Figure 1. The initial input is a complete requirement text. The output is unambiguous requirement texts.

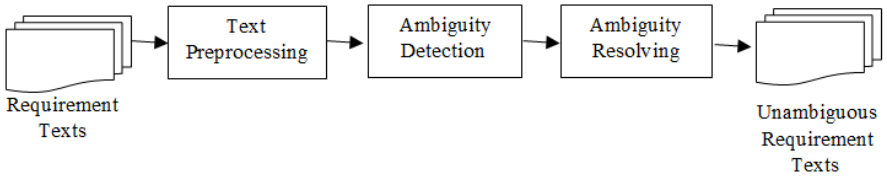


Figure 1DARAArchitecture

The system consists of three major functional process modules

2.1 The Text Preprocessing Module

The Text preprocessing module consists of four stages as shown in Figure 2.

- Sentence splitter: Each sentence is isolatedfrom the input text and is returnedasset of strings.
- Tokenizer: Each sentence is capturedas an input and is separatedinto tokens for examplewords, numbers and punctuation.
- Parts of speech (POS tagger):The words in a documentare determined to a specific part of speech.
- Syntactic parser: sequences of words are changed into structures that show how the sentence's partsconnect to each other. This phaseassists us in recognizing the fundamental parts in each sentence such as subject, object, verb…etc[9].

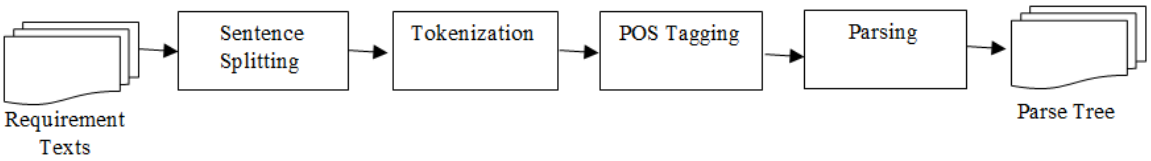


Figure 2 The Text Preprocessing Module

2.2 The Ambiguity Detection Module

This module could apply a several ambiguity measures to a requirement specification to recognize possibly ambiguous sentences. The core goals for this tool are: to detect which sentences in a natural language requirement specification are ambiguous and, for each ambiguous sentence, identify the ambiguity word

and ambiguity type. And calculate the percentage of each ambiguity typein the document. Figure 3 shows The Ambiguity Detection Module architecture.

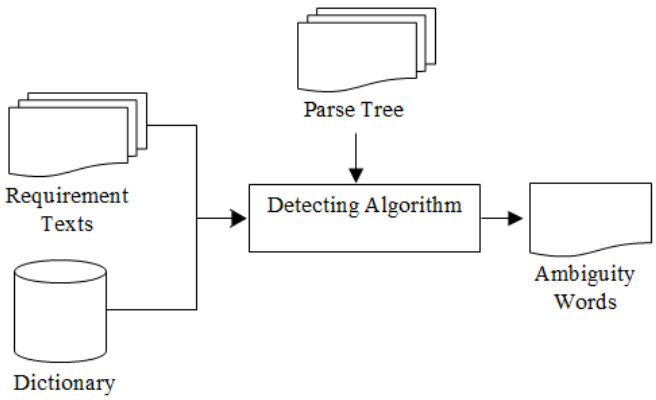


Figure 3The Ambiguity Detection Module

Dictionary is the fundamental element of ambiguity detection whichcontains the ambiguity indicators[10]in the documents. Ambiguous words that outcome from misinterpreted requirements are analyzed and saved into the dictionary. The major goal of this phase is to check and seeif the wordsin software requirements specification document are ambiguous or unambiguous. There are five types of ambiguity Lexical Ambiguity, Referential Ambiguity, Coordination Ambiguity, Scope Ambiguity, Vague. So we identify indicators of each type.

i. Identify Lexical Ambiguity

The Lexical dictionary contains the possible ambiguity indicators such as: bound, break, call, content, continue, contract, count, direct, even, express, form, forward, function, get, job, level, name, notice, number, out, part, position, record, reference, return, set, source, special, standard, string, subject, switch, tail, throw, throw, throw, translate, try, under, value ,and way.

ii. Identify Referential Ambiguity

The Referential dictionarycontains the possible ambiguity indicators such as: I, he, she,it, me, her, him,them,hers, his, its, your, their, our,herself, himself, itself, ours, ourselves, yourself,themselves,yourselves, that, theirs, these, they, this, which, who, you, yours, someone,anyone,everyone,somebody, anybody,everybody,something, anything,and everything.

iii. Identify Coordination Ambiguity

The Coordination dictionarycontains the possible ambiguity indicators such as: and also, and, and/or, but, if and only if, if then, or, and unless.

iv. Identify Scope Ambiguity

The Scope dictionarycontains the possible ambiguity indicators such as: a, all, any, each, few, little, many, much, not, several, and some.

v. Identify Vague

The Vague dictionary contains the possible ambiguity indicators such as: available, common, capability, consistent, easily, easy, effective, efficient, full, general, maximum, minimum, powerful, particular, quickly, random, recent, sufficient, sufficiently, sequential, significant, simple, useful, and various.

2.3 The Ambiguity Resolving Module

Finally, this module focuses in removing and resolving the ambiguity. For each ambiguous sentence, resolve the ambiguity in the sentence automatically as the final step using resolving rules, and therefore improve the natural language requirement specificationdocument. Figure 4 shows The Ambiguity Resolving Module architecture.

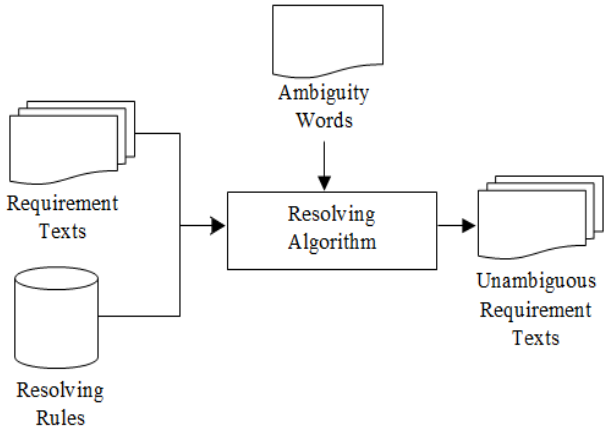


Figure 4The Ambiguity Resolving Module

The resolving ambiguity approach uses the following common rules to check if a sentence contains an ambiguity:

Rule 1: when sentence containing **not only, but also, as well as, both, but, and, and also, or, and/or,X /Y, either, whether, otherwise, meanwhile, whereas, on the other hand** split it to two sentences.

Rule 2: when sentence containing **unless**, replace with **if not**.

Rule 3: when sentence containing **a, an, all, any, some, every, several** replace with **each**.

Rule 4: when sentence containing **should, will, would, may, might, ought to** replace with **shall**.

Rule 5: when sentence containing **There is X in Y, X exists inY** replace with **Y has X**.

Rule 6: when sentence containing **anaphora or pronoun such as they or them** replaces with **the farthest noun.**

Rule 7: when sentence containing **that** replace with **each of which.**

Rule 8: when sentence containing **only, also, almost, even, hardly, just, merely, nearly, and really** put itafterthe first verb**.**

Rule 9: when sentence containing **until, up to, at, during, duration and including, through, by,** or **after** add **only** before it.

Rule 10: when sentence containing **and, or** in same sentence addparentheses.

Rule 11:when sentence containing **many** replace with **each of many**.
Rule 12:when sentence containing **few** replace with **each of few**.
Rule 13:when sentence containing **for up to** replace with **for up to and including**.
Rule 14: when sentence containing **plural nouns** add **each** before it.

3. DARA implementation, results and analysis

DARA was developed using the openNLP and Java language. The Apache OpenNLP library is aJava libraryopen source and machine learning depend on toolkit for the handling of natural language document.OpenNLPsupportsNLP services like sentence segmentation,tokenization, part of speech tagging, parsing, chunking, named entity extraction, and coreference resolution. These services are required to implement more advanced text processing tasks. The OpenNLP library was used to build an effective text processing service. In this section the screenshot of DARA is provided. The graphical user interface is shown to facilitate in the description. Figure 5 show the GUI when the tool is in the run state. The DARA GUI is composed of four principal windows:

- **Input Window**—shows the content of the document file containing the requirements to be detected ambiguity and resolved and analyzed.
- **Output Window**—showsthe detected and resolved software requirement specification.
- **Dictionary Window**—shows the content of the dictionaries and contains function buttons for dictionary handling.
- **Analysis Window**—shows total ambiguities present in the software requirement specification document and percentage ofall ambiguity types and graphical representation.

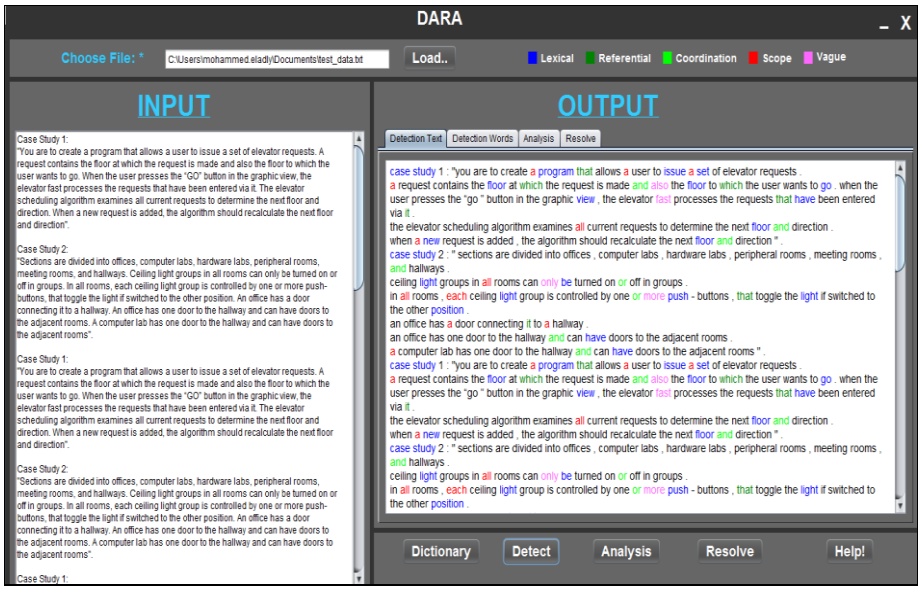


Figure 5 DARA User Interface

3.1 Inputs Data

An analysis of real requirement documents taken from industrial software projects was performed by DARA, in order to test the tool and understand if it may provide a real support to the improvement of the quality of natural language requirements in an industrial environment. The requirement specification documents that were analyzed come from different application domains. The total set of software requirement specification documents is composed of 36 items. The requirements documents were collected from different websites. Number of lines and source of sample software requirement specification documents are presented in Table1.

Table 1Requirements Specification Documents Details

ID	Title	#Sentences	Link
D1	ECMA Standard ECMA-262	34961	http://www.ecma-international.org
D2	FlexRay ™	17133	https://svn.ipd.kit.edu
D3	Landsat Processing System	14726	http://research.it.uts.edu.au
D4	The investigation and control of outbreaks	9440	http://goo.gl/Sv4Ebu
D5	German Health Professional Card	9086	http://www.dkgev.de
D6	Joint Mapping Toolkit	7341	http://research.it.uts.edu.au
D7	VoteCal	3070	http://elections.cdn.sos.ca.gov
D8	Communication Services for DII	2749	http://research.it.uts.edu.au
D9	Foodborn outbreak management	2355	http://goo.gl/pTlgp9
D10	Outbreak management guidelines for healthcare	2099	http://goo.gl/EcYVEi
D11	WHO guidelines for epidemic preparedness	2094	http://goo.gl/PK9yn7
D12	Application to clinical and Public Health	1885	http://goo.gl/hVVy1Y
D13	Document for the Labor Market Information	1856	http://research.it.uts.edu.au
D14	SplitPay	1573	https://www.cise.ufl.edu
D15	Pacemaker	1378	https://svn.ipd.kit.edu
D16	A-7E Avionics System	1339	https://svn.ipd.kit.edu
D17	MODIS Science Data Processing Software	1117	http://research.it.uts.edu.au
D18	PHEMCE strategy	1064	http://goo.gl/hYaipm
D19	Civil Protection Service Resource System	1014	https://svn.ipd.kit.edu
D20	Defense Information Infrastructure	769	http://research.it.uts.edu.au
D21	Coincidence Matrix in the ATLAS Muon	596	http://research.it.uts.edu.au
D22	Post Grass System	516	http://research.it.uts.edu.au
D23	Light Control System	427	http://research.it.uts.edu.au
D24	E-Store Project	419	https://www.utdallas.edu
D25	University Library Information System	408	https://svn.ipd.kit.edu
D26	Developing a management system	401	http://goo.gl/0l5sth
D27	Ludo	398	https://svn.ipd.kit.edu
D28	Whois Protocol	187	http://www.ietf.org
D29	Display Management System	91	https://svn.ipd.kit.edu
D30	Cable TV Package Purchase	79	https://svn.ipd.kit.edu
D31	ATM Simulation	33	http://nlrp.ipd.kit.edu
D32	Sogno Hotel Reservation Service	24	https://svn.ipd.kit.edu
D33	Library	18	http://nlrp.ipd.kit.edu
D34	Ambulance Despatching System	17	https://svn.ipd.kit.edu
D35	Address Book	14	http://nlrp.ipd.kit.edu
D36	Mellor's Steam Boiler	7	http://nlrp.ipd.kit.edu

3.2 Outputs Data

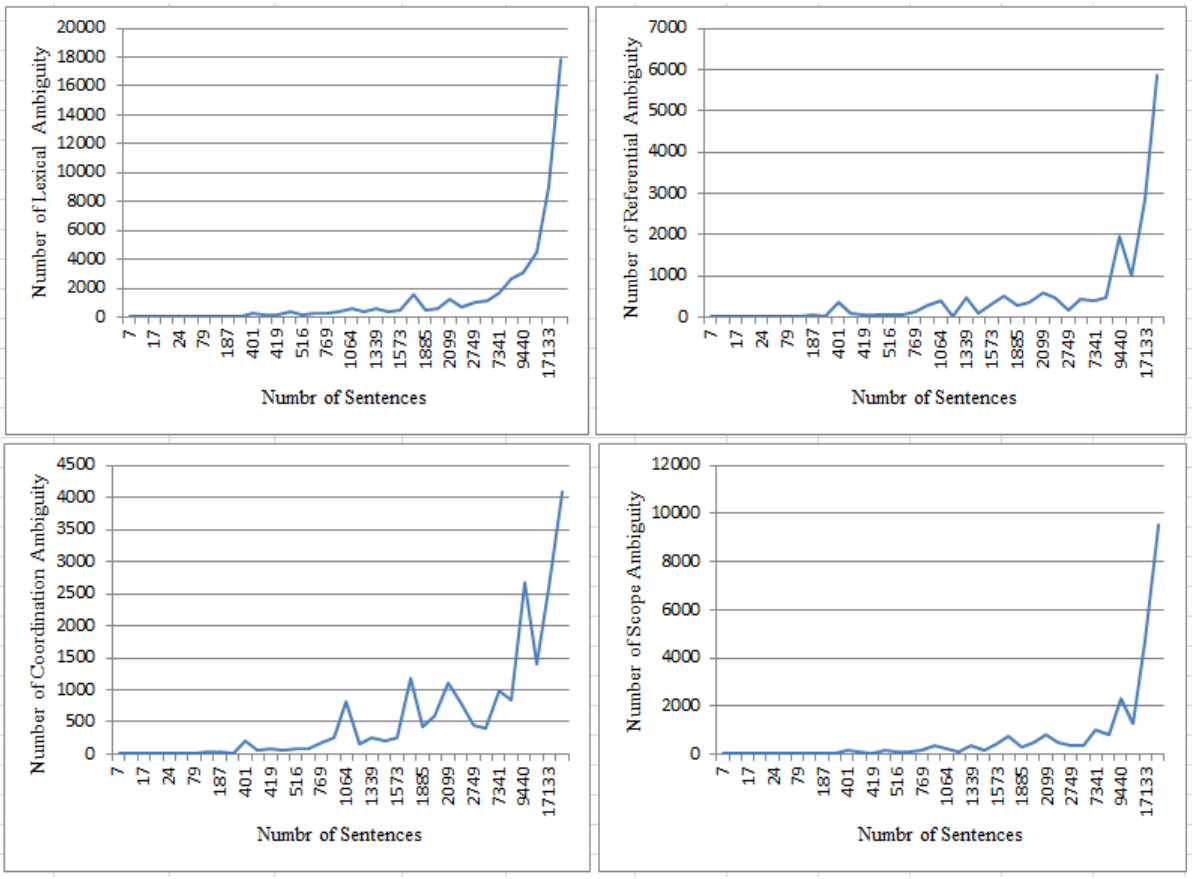
The results of this type of validation have been very interesting.They are presented in the Table 2that shows, for each evaluated document, the number of indicators' occurrencesof all the datasets. It displays the total numbers of ambiguities occur in software requirement specification documents and percentages of lexical,referential, coordination,scope and vague ambiguity for each document of software requirement specification. All experiments were executed on a 2.3 GHz Intel Core i5 with 4 GB of memory.

Table 2The Occurrences of The Possible Ambiguities for each Indicator

ID	Number of Lexical Ambiguity	Number of Referential Ambiguity	Number of Coordination Ambiguity	Number of Scope Ambiguity	Number of Vague Ambiguity	Number of Detected Sentences	Number of Resolved Sentences	Time
D1	38.8%	12.7%	8.8%	20.6%	19.1%	56.4%	44.6%	24min
D2	37.6%	11.8%	10.8%	19.6%	20.3%	58.4%	55.6 %	8 min
D3	47.8%	10.9%	14.9%	13.4%	13.1%	35.8%	46.2%	4 min
D4	22.8%	14.2%	19.5%	16.9%	26.5%	65.2%	62.3%	4 min
D5	44.0%	8.1%	13.8%	13.1%	21.0%	36.5%	42.7%	4 min
D6	33.7%	7.5%	19.6%	20.2%	19.0%	32.8%	65.0 %	2 min
D7	36.0%	13.0%	12.2%	11.7%	27.1%	38.3%	50.3 %	1min
D8	43.2%	7.5%	18.6%	14.2%	16.4%	49.2%	49.9%	1 min
D9	20.7%	13.0%	22.3%	14.3%	29.7%	68.2%	73.4 %	1 min
D10	24.3%	12.3%	22.4%	16.8%	24.2%	72.3%	73.4 %	1 min
D11	22.4%	13.0%	21.6%	17.8%	25.2%	59.2%	69.9%	1 min
D12	25.1%	14.2%	22.5%	14.8%	23.4%	57.2%	57.8%	53 sec
D13	28.1%	9.4%	20.8%	13.7%	28.0%	76.8%	83.6 %	55 sec
D14	28.2%	16.0%	13.4%	21.7%	20.8%	52.3%	67.4 %	32 sec
D15	36.4%	7.9%	18.3%	16.8%	20.6%	43.0%	58.8%	36 sec
D16	28.9%	23.2%	12.4%	18.1%	17.4%	65.0%	61.4%	34 sec
D17	40.7%	4.3%	19.7%	8.7%	26.7%	47.9%	38.9%	48 sec
D18	21.4%	14.4%	29.7%	9.0%	25.6%	63.9%	75.0 %	50 sec
D19	23.8%	17.3%	15.2%	19.8%	24.0%	73.1%	62.6%	46 sec
D20	26.8%	14.4%	22.2%	18.7%	17.9%	51.4%	67.6%	52 sec
D21	38.3%	10.6%	11.7%	12.0%	27.5%	55.4%	50.3%	30 sec
D22	30.2%	12.2%	15.0%	18.2%	24.4%	57.0%	54.4%	19 sec
D23	47.1%	9.3%	9.7%	21.1%	12.8%	65.6%	62.9%	23 sec
D24	31.9%	16.8%	18.0%	15.1%	18.3%	39.1%	58.5%	18 sec
D25	29.7%	16.8%	13.3%	18.7%	21.4%	61.0%	55.8%	19 sec
D26	20.5%	31.0%	18.0%	11.6%	18.9%	80.5%	73.1%	30sec
D27	43.7%	13.2%	6.8%	21.6%	14.7%	18.3%	53.4%	17sec
D28	18.9%	21.4%	19.9%	20.9%	18.9%	51.3%	66.7%	40 sec
D29	26.7%	16.3%	23.0%	12.6%	21.5%	54.9%	72.0%	17sec
D30	24.2%	17.5%	11.7%	17.5%	29.2%	51.9%	82.9%	22sec
D31	23.5%	8.2%	12.4%	30.0%	25.9%	84.8%	96.4%	32sec
D32	27.9%	17.6%	11.8%	26.5%	16.2%	100.0%	79.2%	28sec
D33	28.8%	1.7%	20.3%	39.0%	10.2%	88.9%	100.0%	12sec
D34	13.7%	9.8%	5.9%	47.1%	23.5%	76.5%	84.6%	18sec

D35	40.8%	9.9%	12.0%	21.8%	15.5%	85.7%	100.0%	22sec
D36	45.0%	2.5%	12.5%	27.5%	12.5%	100.0%	100.0%	25sec

Figure 6 shows that some particular ambiguities are more frequently detected than others by DARA. Especially lexical, scope and vague ambiguity seems to be the types of ambiguityimpacting in a large part of the requirement sentences of documents. Figure 6 shows that document 3 demonstrate a decrease in percentage distribution of all ambiguity types detected because of the document domain(Document 3 about satellite) and it shows that document 26 demonstrate an increase in percentage distribution of all ambiguity types detected because of the document domain covered in dictionaries.



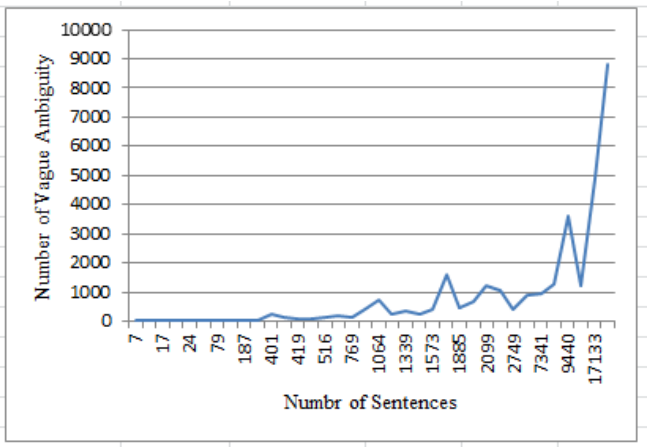


Figure 6 Percentage distribution of all ambiguity types detected

Figure 7 shows the percentage of each type of ambiguities is detected by DARA.The outcomes of the use of DARA on these 36 case studies show that the occurrences of the possible ambiguities are significantly high around 60% of the total number of requirements sentences (lexical ambiguity 37%, referential ambiguity 9%, coordination ambiguity 13%, scope ambiguity 25% and vague 16%).Figure 8 shows the numbers of detected and resolved sentences by DARA. The outcomes of the use of DARA on these 36 case studies show that the number of detected sentences that have possible ambiguities and the number of solved sentences are increased when the number of sentences increased. DARA solve 67% of ambiguities in the total number of requirements sentences.

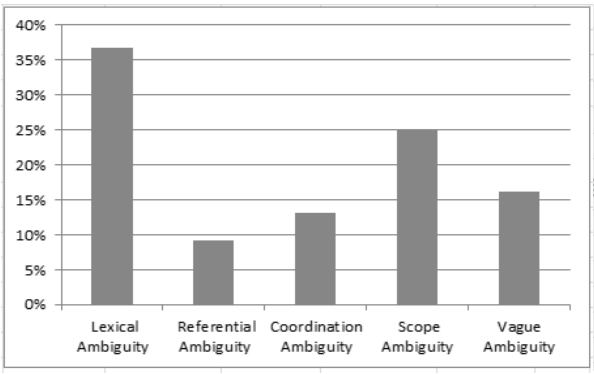


Figure 7 Percentage distribution of ambiguity types detected

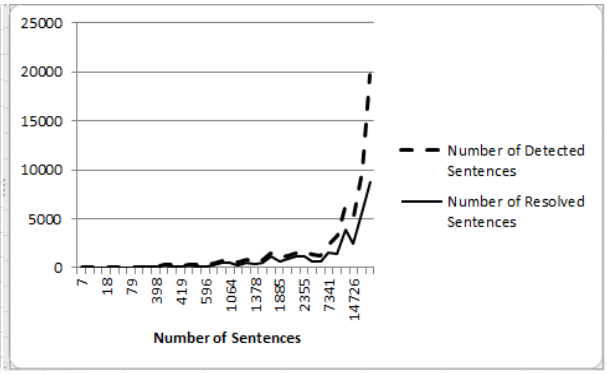


Figure 8 Percentage distributions of detected and resolved sentences

4. Conclusion

We have discussed that ambiguity is common in natural language requirements. When different stakeholders understand the same text differently, system incorrectly implementedriskbeing high [11]. So we implement DARA to detect and resolve ambiguities.According to the defined approach, DARA doesn't force the requirement engineers to follow a particular standard or style in writing [12].We execute our approach on 36 case studies. DARA can detect lexical ambiguity, referential ambiguity, coordination ambiguity, scope ambiguity, vague. It can measure the percentage of each ambiguity type. It can solve 67% of ambiguities in NL requirements specification documents. In this paper,by using a rule based approach

we proved that it is possible to identify and resolve ambiguity automatically in natural language requirements. We employed algorithm to recognize ambiguities from sentences using dictionaries. Our aim is to enhance the requirements quality by assist requirements analysts to detect and resolvepossible ambiguity requirements. In future work, we will try to extend our work to convert requirements specification documents to UML diagrams.

References

[1] Basili, Victor R., Scott Green, Oliver Laitenberger, FilippoLanubile, Forrest Shull, SivertSorumgard. 1995. The Empirical Investigation of Perspective-Based Reading. Technical report the empirical investigation of perspective based reading(pp. 133-164).
[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, ProceedingsIEEE, Annual (pp. 15-21).
[4] Christel, M. G., & Kang, K. C. 1992. Issues in requirements elicitation (No.CMU/SEI-92-TR-12). CARNEGIE-MELLONUNIV 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, http://se.uwaterloo.ca/˜dberry/handbook/ambiguityHandbook.pdf.
[7] 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.
[8] A. Fantechi, S. Gnesi, G. Lami, and A. Maccari,2003.Applications of Linguistic Techniques for Use CaseAnalysis," Requirements Engineering, vol. 8, (pp. 161-170).
[9] 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, (pp. 2-40).
[10] Sri Fatimah Tjong, 2008, Avoiding ambiguity in requirements specifications.Thesis submitted to the University of Nottingham for the degree of Doctor of Philosophy.
[11] Sven Körner and TorbenBrumm, 2009, RESI-A natural language specification improver. IEEE International Conference on Semantic Computing (ICSC),(pp. 1-8).
[12] Sommerville, I. and Sawyer, P, 1997,Requirements engineeringA good practice guide. Chichester: John Wiley & Sons Ltd.