An Experience with the Application of Three NLP Tools for the Analysis of Natural Language Requirements

Monica Arrabito[1], Alessandro Fantechi[2,3](✉), Stefania Gnesi[3], and Laura Semini[1,3]

[1] Dipartimento di Informatica, Università di Pisa, Pisa, Italy
monica.arrabito@hotmail.it, laura.semini@unipi.it

[2] Dipartimento di Ingegneria dell'Informazione, Università di Firenze, Florence, Italy
alessandro.fantechi@unifi.it

[3] Istituto di Scienza e Tecnologie dell'Informazione "A.Faedo" Consiglio Nazionale delle Ricerche, ISTI-CNR, Pisa, Italy
stefania.gnesi@isti.cnr.it

Abstract. We report on the experience made with three Natural Language Processing analysis tools, aimed to compare their performance in detecting ambiguity and under-specification in requirements documents, and to compare them with respect to other qualities like learnability, usability, and efficiency. Two industrial tools, Requirements Scout and QVscribe, and an academic one, QuARS, are compared.

Keywords: Natural Language Processing · Natural language requirements · Ambiguity

1 Introduction

Natural language (NL) is the most common way to express software requirements even though it is intrinsically ambiguous, and ambiguity is seen as a possible source of problems in the later interpretation of requirements. Ambiguity is one of the most difficult defects to avoid since natural language is ambiguous by nature and devoid of formal semantics. The lack of intrinsic formalism of the requirements document must therefore be compensated by a detailed analysis during the initial stages of the product life cycle in order to correctly extrapolate all the needed information. For this reason, part of the work carried out during the analysis phase is intended for disambiguation of the requirements.

Natural Language Processing (NLP) techniques have been used to analyse requirement documents to single out, among other issues, ambiguity and under-specification defects, analyzing the structure of sentences using grammatical and lexical analysis, dictionaries and parsers for natural language [1].

Work partially funded by MIUR project PRIN 2017FTXR7S *IT MaTTerS* (Methods and Tools for Trustworthy Smart Systems).

Recently, several tools have been developed for analyzing NL requirements in a systematic and automatic way by means of NLP techniques with a focus on ambiguity detection.

In this paper, we report on the experience made with thee NLP analysis tools, aimed to compare their performance in detecting ambiguity and under-specification, and to compare them with respect to other qualities like learnability, usability, efficiency. Two industrial tools, Requirements Scout and QVscribe, and an academic one, QuARS, are analysed.

2 The Scope of the Experience

The choice of the three mentioned tools can be traced back to their similarities:

– they perform an automatic detection of possible language defects that may determine interpretation problems and affect the following development stages;
– they highlight the word or construct that they reveal as defective;
– the detected defects may however be false positives, and a subsequent manual analysis is needed.

The limited scope of our experience on the one hand allows us to focus on similar tools to better highlight the differences, on the other hand it is a threat to the validity of our results. To broaden the investigation, other requirement analysis tools (see for instance Sect. 2.4) can be considered in a future work.

Notwithstanding its limited extension, we believe that this study can provide some useful insights on the current state of the art of automatic detection of ambiguity in natural language requirements.

In the following, we introduce the NLP tools we use for the comparison: QuARS, QVscribe, and Requirements Scout.

2.1 QuARS

QuARS - Quality Analyzer for Requirement Specifications - is a tool for analyzing NL requirements in a systematic and automatic way by means of NLP techniques with a focus on ambiguity detection [5].

QuARS performs an automatic linguistic analysis of a requirements document in plain text format, based on a given quality model. Its output indicates the defective requirements and highlights the words that reveal the defect.

Below, we present the indicators used by QuARS to detect defects of lexical and syntactic ambiguity in NL sentences.

Defect	Indicators
Vagueness	Dictionary: clear, easy, strong, good, bad, adequate...
Subjectivity	Dictionary: similar, have in mind, take into account, ...
Optionality	Dictionary: or, and/or, possibly, eventually, case, if possible, if appropriate...
Weakness	Dictionary: can, could, may, ...
Implicity	Demonstrative adjectives or pronouns
Under-specification	Wordings missing a qualification (e.g.: interface or information without a qualifier, such as user and price, respectively)
Multiplicity	Multiple syntactic constructs such as multiple verbs or multiple subjects

The defect identification process is divided into two, independent, parts. The first part, *lexical analysis*, detects candidate defective terms using a set of dictionaries. Lexical analysis permits to capture *optionality, subjectivity, vagueness, optionality, and weakness* defects. The second part is *syntactical analysis*, that captures *implicity, multiplicity and under-specification* defects.

Other features of QuARS are (i) metrics derivation for evaluating the quality of NL requirements; (ii) the capability to modify existing dictionaries, and to add new dictionaries for new indicators; (iii) view derivation, to identify and collect together those requirements belonging to given functional and non functional characteristics, by defining specific requirements.

2.2 QVscribe

QVscribe [6] is an industrial tool for requirements analysis for quality and consistency, developed by QRA (https://qracorp.com/). QVscribe analyzes the quality of the requirements, highlighting ambiguity, inconsistencies and possible similarities, providing scores to the single requirements and to the whole document. It generates a detailed report that can be used to increase the quality of requirements, reducing the review and rewriting work.

The analysis performed by QVscribe is based on part of the rules defined by the INCOSE Guide for Writing Requirements. The defects detected by the tool and the related indicators can be classified according to following table.

Defect	Indicators
Imperatives	Absence, negation, or multiple occurrence of imperatives
Optional escape clauses	Optional terms like: possibly, may, ...
Vague words	Vague nouns and verbs as: various, completed, ...
Cross-referencing pronouns	Both, everybody, anyone, it, ...
Non-specific temporal words	Early, years ago, before, ...
Continuances	Otherwise, in particular, below, following, ...
Superfluous infinitives	Since they can hide the subject, as in: shall permit
Passive voice	Since it can hide the subject, ex: based, found, shipped
Immeasurable quantification	Abundant, far, always, all, ...
Incomplete sentences	Missing critical details of who must do something or what must be done

2.3 Requirements Scout

Requirements Scout [3] is developed by Qualicen GmbH, to analyze requirements specifications (https://www.qualicen.de/en/). It is distributed as pluging of NL editors, including Microsoft Word, which makes is suitable for immediate feedback when writing requirements. Requirements Scout operates similarly to QuARS and QVscribe, using dictionaries and syntactic rules to specify the critical words that might denote a quality smell. We list below defects and indicators.

Defect	Indicators, if any, or motivation
Long&complicated sentences	Which are difficult to read and prone to ambiguities
Passive voice	Done, found, sent, ... since they can hide the subject
Multiple negations	Requirements must be expressed in positive terms
Universal quantifiers	All, always, every, any, nothing, ...
Imprecise phrases (vagueness)	Possibly, various, current, small, general, if possible, ...
Vague pronouns	That, which, their, it, nobody, ...
Comparatives & superlatives	Faster than, fastest, bigger than, ... they make a requirement not understandable in isolation
Exactly one shall or should	More than one occurrence of shall or should
Occurrence of will or may	Weak verbs such as will, may, ...
Wrong abstraction level	To exclude implementation details
Dangerous slash	"/" , that can be interpreted both as an *and* and an *or*
UI details	Requirements should not contain details of the user interface.
Cloning	Since duplicates burden successive maintenance

Besides identifying the defects, it also permits to keep track of different versions of the requirements, creating a complete history of the detected defects: as soon as the requirements are updated, the tool re-analyzes the modified parts and shows whether the update has eliminated existing defects or has introduced new ones.

2.4 Other Tools

A recent industrial tool for assisting the analyst in the definition of NL requirements is ReqSuite, by OSSENO Software GmbH. Differently than the tools considered in this paper, which highlight defective words or constructs, ReqSuite supports rigorous requirement definition by correcting the writer according to some patterns and hence is out of the scope for our study.

Other candidate tools to experiment with are RAT (Requirements Authoring Tool) from REUSE (https://www.reusecompany.com/) and IBM RQA (https://www.ibm.com/products/requirements-quality-assistant), both able to detect ambiguities, but those, for their commercial nature, were not immediately available for our study.

An alternative to the experimentation of off-the-shelf tools is the adoption and customization of more general and flexible NLP tools, that allow to tune the kind of detected ambiguities and other defects. GATE [2] is an example of such tools: it collects several NLP modules and provides a means to define ad hoc rules (JAPE rules), so to create advanced and customized NLP solutions. As an example related to requirement analysis, in [4] GATE was used to tune the proposed requirement analysis according to the requirement writing style adopted by the involved company, achieving a significantly better quality of the analysis.

3 Application of the Tools to a E-Shop Case Study

We report our observations when applying QuARS, QVscribe and Requirements Scout to a running example, a simple E-shop (requirements in Table 1). The tools are analysed first for their general qualities, then in their ability to support ambiguities detection in requirements.

3.1 General Qualities Evaluation

We first address *documentation, learnability*, and *usability*. QuARS was simple to learn and use without referring to any manual. QVscribe comes equipped with good documentation and video tutorials and it was easy to be acquainted with. Requirements Scout is the tool were most difficulties were encountered, because of lack of documentation, a non intuitive interface, and a complex setup of the user profile. To give a rough measure of learnability, we report the number of hours of training in order to proficiently use them: 30 for QuARS, 36 for QVscribe, 48 for Requirements Scout.

Table 1. Requirements of the E-shop case study

R1	The system shall enable the user to enter the search text on the screen
R2	The system shall display all the matching products based on the search
R3	The system possibly notifies with a pop-up the user when no matching product is found on the search
R4	The system shall allow a user to create his profile and set his credentials
R5	The system shall authenticate user credentials to enter the profile
R6	The system shall display the list of active and/or the list of completed orders in the customer profile
R7	The system shall maintain customer email information as a required part of customer profile
R8	The system shall send an order confirmation to the user through email
R9	The system shall allow an user to add and remove products in the shopping cart
R10	The system shall display various shipping methods
R11	The order shall be shipped to the client address or, if the "collect in-store" service is available, to an associated store
R12	The system shall enable the user to select the shipping method
R13	The system may display the current tracking information about the order
R14	The system shall display the available payment methods
R15	The system shall allow the user to select the payment method for order
R16	After delivery, the system may enable the users to enter their reviews or ratings
R17	In order to publish the feedback on the purchases, the system needs to collect both reviews and ratings
R18	The "collect in-store" service excludes the tracking information service

Efficiency was found to be an issue for Requirements Scout – to analyze documents of 20 and 50 requirements the tool takes 1 min and 2 min respectively – while it was not for QVscribe and QuARS: with both tools the analysis time for the considered documents was few seconds. The problem was probably due to the larger amount of checks performed by Requirements Scout, so it has to be considered as an issue related to the particular usage of the tool for ambiguity detection, rather than a generic low performance of the tool.

Extensibility is represented by the ability to add new quality indicators. QuARS has this feature and it also permits the user to select the indicators she wants to use for the analysis. In QVscribe, only the modification of the indicators already present in the tool is permitted, by adding or removing terms to be identified during the analysis. Requirements Scout implements indicators' selection too, but indicators are fixed.

Report generation is possible in QuARS and QVscribe. In QuARS the report contains, for each quality indicator, the list of requirements that present an ambiguity, together with the terms deemed incorrect. The report generated by QVscribe assigns to each requirement a score ranging from 1 to 5, depending on the gravity of the defects. Results can be filtered to focus on specific defects.

Other qualities are *interoperability* and *version control.* A particularly important positive aspect of QVscribe is the possibility of integrating the tool as a Word feature, so that the analysis of a document can be started by selecting QVscribe from the Word ribbon, and selecting the requirements to be analyzed.

A version control system is offered by Requirements Scout: the tool records the history containing the various versions of a document so that the comparison of two versions of the same document returns the list of defects added or removed. However the tool does not permit any editing of the document under analysis: the user has to edit the document externally and load the new version.

3.2 Evaluation of the Ability to Detect Defects in the Requirements

We now focus on analysing the three tools from the point of view of the ability of their indicators to detect ambiguities and under-specifications. We report in Table 2 the raw outcomes of the analysis of the E-shop example with the three tools, requirement by requirement: the "Indicator" column shows the words that have been considered by each tool to indicate a certain defect (reported in the last column). The detected defects results have then been manually analysed to distinguish real defects from false positives. The detailed results of this analysis are discussed in the following indicator by indicator. Table 3 cumulatively shows the number of *false positives* and *ambiguities*, as the result of the manual analysis of the tools' outcome of Table 2.

For *vagueness* QuARS detects a defect, QVscribe and Requirements Scout detect four defects each. The vagueness related to requirement **R10**, detected both by QuARS and Requirements Scout, can be indeed classified as a real defect (*various*). The same happens for the term *possibly* detected by Requirements Scout in **R3**. All the other defects are false positives.

We note that the term *possibly* in QuARS and QVscribe is an indicator of Optionality and is hence detected according to another indicator.

With respect to *optionality*, we refer to its meaning as in QuARS, and include the term *possibly*, classified as Optional Escape Clause by QVscribe. According to this indicator, there are four ambiguities detected by QuARS (**R3**, **R6**, **R11**, and **R16**) and one by QVscribe (**R3**). Optionality is not an indicator of Requirement Scout. The good number of defects detected by QuARS is due to fact that it is the only tool looking for occurrences of *or* and of *and/or*.

For *weakness* all the tools perform the same on E-shop. Weakness is referred to as *optional escape clause* in QVscribe and *occurrence of will or may* in Requirement Scout. When applying the tools to other documents, we have also observed that QuARS and QVscribe detect the weak verb *can* which is not detected by Requirements Scout.

The only two defecs related to *multiplicity* in QuARS are indeed disjunctions (**R11, R16**) that were already detected by *optionality* indicators.

Table 2. Results of the application of QuARS, QVscribe, and Requirements Scout to the e-shop case study. Requirements **R5**, **R8**, and **R14** contain no defect according to all tools.

Requirement	Tool	Indicator	Defect
R1 The system shall enable the user to enter the search text on the screen	**QuARS**	–	Multiplicity
	QVscribe	Enable	Vague words
	Req. Scout	Screen	UI details
R2 The system shall display all the matching products based on the search	**QuARS**		
	QVscribe	All	Universal quantifiers
		Based	Passive voice
	Req. Scout	All	Universal quantifiers
R3 The system possibly notifies with a pop-up the user when no matching product is found on the search.	**QuARS**	Possibly	Optionality
		–	Multiplicity
	QVscribe	Possibly	Optional escape clauses
		When	Immeasurable quantification
		No	Universal quantifiers
		Found	Passive voice
		–	No imperatives
	Req. Scout	Possibly	Vagueness
		Found	Passive voice
		–	Exactly one shall or should
R4 The system shall allow a user to create his profile and set his credentials	**QuARS**	–	Multiplicity
	QVscribe	Allow	Superfluous infinitives
		His	Cross-referencing pronouns
	Req. Scout	His	Vague pronouns
R6 The system shall display the list of active and/or the list of completed orders in the customer profile	**QuARS**	And/or	Optionality
	QVscribe		
	Req. Scout	Completed	Vagueness
		And/or	Dangerous slash
R7 The system shall maintain customer email information as a required part of customer profile	**QuARS**	–	Multiplicity
	QVscribe	Maintain	Superfluous infinitives
		As	Immeasurable quantification
	Req. Scout		
R9 The system shall allow an user to add and remove products in the shopping cart	**QuARS**	–	Multiplicity
	QVscribe	Allow	Superfluous infinitives
	Req. Scout	–	–
R10 The system shall display various shipping methods	**QuARS**	Various	Vagueness
	QVscribe	–	–
	Req. Scout	Various	Vagueness
R11 The order shall be shipped to the client address or, if the “collect in-store” service is available, to an associated store	**QuARS**	–	Multiplicity
		Or	Optionality
	QVscribe	Shipped	Passive voice
	Req. Scout	Shipped	Passive voice
R12 The system shall enable the user to select the shipping method	**QuARS**	–	Multiplicity
	QVscribe	Enable	Vague words
	Req. Scout	–	–

(*continued*)

Table 2. (*continued*)

Requirement	Tool	Indicator	Defect
R13 The system may display the current tracking information about the order	**QuARS**	May	Weakness
	QVscribe	May	Optional escape clauses
		About	Vague words
		–	No imperatives
	Req. Scout	Current	Vagueness
		May	Occurrence of will or may
		–	Exactly one shall or should
R15 The system shall allow the user to select the payment method for order	**QuARS**	–	–
	QVscribe	Allow	Superfluous infinitives
	Req. Scout	–	–
R16 After delivery, the system may enable the users to enter their reviews or ratings	**QuARS**	–	Multiplicity
		Or	Optionality
		May	Weakness
	QVscribe	After	Non-specific temporal words
		May	Optional escape clauses
		Enable	Vague words
		Their	Cross-referencing pronouns
	Req. Scout	–	No imperatives
		Their	Vague pronouns
		May	Occurrence of will or may
		–	Exactly one shall or should
R17 In order to publish the feedback on the purchases, the system needs to collect both reviews and ratings	**QuARS**	–	–
	QVscribe	Both	Cross-referencing pronouns
		–	No imperatives
	Req. Scout	–	Exactly one shall or should
R18 The "collect in-store" service excludes the tracking information service	**QuARS**	–	–
	QVscribe	–	No imperatives
	Req. Scout	–	Exactly one shall or should

Looking at Table 3, we notice that the absence of imperatives is a main ambiguity indicator. This is an indicator considered by QVScribe (*no imperatives)* and in Requirement Scout (*exactly one shall or should*), but not by QuARS. However, we can notice that the requirements lacking an imperative and being defective (**R3, R13, R16**) are those containing terms such as *if possible*, or weak verbs such as *may* or *can*. QuARS captures them with other indicators, namely *optionality, weakness* and *cross-tree constraints* indicators.

Table 3. Summary of ambiguity detection (n.a. means not applicable)

E-shop	QuARS		Qvscribe		Requirements Scout	
	F.Pos.	*Amb.*	*F.Pos.*	*Amb.*	*F.Pos.*	*Amb.*
Vagueness	-	1	4	-	2	2
Optionality	-	4	-	1	n.a.	
Weakness	-	2	-	2	-	2
Multiplicity	6	2	n.a.		n.a.	
Under-Specificaiton	-	-	n.a.		n.a.	
Imperatives	n.a.		2	3	2	3
Vague Pronouns	n.a.		1	2	-	2
Passive voices	n.a.		1	2	-	2
Immeasurable quantification	n.a.		2	2	1	-
Superflous infinitives	n.a.		4	-	n.a.	
Incomplete sentences	n.a.		-	-	n.a.	
Long/complicated sentences	n.a.		n.a.		-	-
Multiple negations	n.a.		n.a.		-	-
Comparatives, superlatives	n.a.		n.a.		-	-
Wrong abstraction level	n.a.		n.a.		-	-
Dangerous slash	n.a.		n.a.		-	1
UI details	n.a.		n.a.		1	-

4 Conclusions

The three tools examined have shown to be comparable in all the considered dimensions. They apparently use different indicators but in the end (e.g. weak verbs vs no imperatives) they find roughly the same defects. Best performances are obtained with best dictionaries, which means that there is room for lowering the false negative rate with better dictionaries: to this end the extensibility features of QuARS, permitting to add and modify dictionaries and of QVscribe that permits to modify the built-in dictionaries are suited and helpful.

There are some differences when considering other quality aspects, and the outcome of the comparison can help the vendors to refactor their tool and beat the competitors. However, in the end, they all share a similar defect detecting strategy, and from this respect younger tools (QVscribe and Requirements Scout) do not perform better than QuARS developed 20 years before.

References

1. Arora, C., Sabetzadeh, M., Briand, L., Zimmer, F.: Automated checking of conformance to requirements templates using natural language processing. IEEE Trans. on Softw. Eng. **41**(10), 944–968 (2015)

2. Cunningham, H.: Gate, a general architecture for text engineering. Comput. Humanit. **36**(2), 223–254 (2002)
3. Femmer, H.: Requirements quality defect detection with the Qualicen requirements Scout. In: Joint Proceedings of 23rd International Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2018). CEUR Workshop Proceedings, vol. 2075. CEUR-WS.org (2018)
4. Ferrari, A., Gori, G., Rosadini, B., Trotta, I., Bacherini, S., Fantechi, A., Gnesi, S.: Detecting requirements defects with NLP patterns: an industrial experience in the railway domain. Empirical Softw. Eng. **23**(6), 3684–3733 (2018). https://doi.org/10.1007/s10664-018-9596-7
5. Gnesi, S., Lami, G., Trentanni, G.: An automatic tool for the analysis of natural language requirements. Comput. Syst. Sci. Eng. **20**(1), 1–17 (2005)
6. Kenney, O., Cooper, M.: Automating requirement quality standards with QVscribe. In: Joint Proceedings of the 26th International Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2020). CEUR Workshop Proceedings, vol. 2584. CEUR-WS.org (2020)