A comparison of NLP Tools for RE to extract Variation Points

Monica Arrabito
Dipartimento di Informatica, Università di Pisa, monica.arrabito@hotmail.it

Alessandro Fantechi
Dip. di Ingegneria dell'Informazione, Università di Firenze, alessandro.fantechi@unifi.it

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

Laura Semini
Dipartimento di Informatica, Università di Pisa, laura.semini@unipi.it

Abstract

In the requirement engineering of software product lines, several researches have focused on exploiting NLP techniques and tools to extract information related to features and variability from requirement documents. In a previous work we have proposed the use of the tool QuARS, a NLP Tool for Requirements Analysis, showing that some of the indicators used to detect NL ambiguity can also be exploited to detect variability. In this paper we discuss a comparison of the application at this regard of QuARS and other Requirements Analysis tools presented in the last edition of NLP4RE, in particular with respect to their ability to extract potential variation points, in search of better performances and of novel indicators.

NLP, natural language requirements, variability, 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. Natural Language Processing (NLP) techniques have been used to analyse requirement documents to single out, among other issues, ambiguity defects.

Although ambiguity or under-specification at requirement level are usually seen as defects, they can be sometimes deliberate, as a form to leave some issues to further analysis or to later decision. In this case, ambiguity

In: M. Sabetzadeh, A. Vogelsang, S. Abualhaija, M. Borg, F. Dalpiaz, M. Daneva, N. Fernández, X. Franch, D. Fucci, V. Gervasi, E. Groen, R. Guizzardi, A. Herrmann, J. Horkoff, L. Mich, A. Perini, A. Susi (eds.): Joint Proceedings of REFSQ-2020 Workshops, Doctoral Symposium, Live Studies Track, and Poster Track, Pisa, Italy, 24-03-2020, published at http://ceur-ws.org

Defect	**Indicators**
vagueness	*clear, easy, strong, good, bad, adequate...*
subjectivity	*similar, have in mind, take into account,...*
optionality	*or, and/or, possibly, eventually, case, if possible, if appropriate...*
implicity	demonstrative adjectives or pronouns
weakness	weak verbs (e.g.: *can, could, may*)
under-specification	wordings to be instantiated (e.g.: *information, interface*)
multiplicity	multiple syntactic constructs such as multiple verbs or multiple subjects

Variability	**Indicators**
variability	*if, where, whether, when, choose, choice, configurable...*
cross-tree constraints	*need, request, require, excludes, expect*

Table 1: Defects and variability detected by QuARS

defects could give an indication of possible variability, either in design choice, in implementation choices or by later configurability.

In this view, ambiguous or under-specified requirements can be considered as requirements for a *product line* [PBvdL05], where different products can be obtained by different choices in design, implementation and configuration. So, analysing a requirement document, the ambiguity that is detected can be used to identify possible *variation points*, where ambiguity is not a defect but points to different choices that can give space for a range of different products.

In [FGS17] we have presented this idea, with its consequence, specifically that NLP tools may help in revealing information on variability that can be later used to define feature models from which different systems can be instantiated. In subsequent works [FFGS18a, FFGS18b] we presented the results of an empirical evaluation of the position, performed on large requirement documents from industry using QuARS [GLT05]. 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. The evaluation has shown that some ambiguous terms, such as “and/or” “or”, and weak terms such as “may” or “could” are more likely to indicate variability rather than ambiguity. Instead, typically vague terms, such as “useful”, “significant”, etc. are more likely to indicate ambiguity. This has been realized implementing a quality model composed of high level quality properties for NL requirements to be evaluated by means of indicators directly detectable and measurable on NL requirement documents.

Here, we report on the experience made with industrial NLP analysis tools, aimed to compare their performance in detecting variability with that of QuARS, and to see whether they include any fresh indicator relevant to detect variability [Arr19]. We have considered the tools that were presented in a showcase at NLP4RE19 [DFF+19], namely Requirements Scout, Semantic Processing Platform, QVscribe, and ReqSuite. After a preliminary analisys, we restricted the attention to Requirements Scout and QVscribe and we have run experiments to compare them with QuARS. Our aim is twofold: to validate the approach we proposed in [FGS17] and to compare the effectiveness of the different indicators to extract variability from linguistic defects.

In Section 2 we introduce the tools used for the comparison, in Section 3 we describe the results of the application of the tools to a case study and analyse the results. A related works and conclusions section follows.

2 The tools

2.1 QuARS

QuARS allows to perform an initial parsing of the requirements for automatically detecting potential linguistic defects due to ambiguity that can determine interpretation problems at the subsequent development stages of a software product. The analysis is done starting from a quality model composed of high level quality properties for NL requirements to be evaluated by means of indicators. These indicators are listed in the first part of Table 1. The second part of the table includes the specific variability revealing indicators introduced in [FFGS18b, AFGS20].

2.2 QVscribe

QVscribe is a tool for requirements analysis for quality and consistency, developed by QRA (https://qracorp.com/). QVscribe analyzes the quality of the requirements, highlighting ambiguity, inconsis-

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,...* e.g. an office has a door connecting *it* to a hallway
non-specific temporal words	*early, years ago, before, ...*
continuances	*otherwise, in particular, below, following, ...*
superfluous infinitives	as in *shall permit* since they can hide the subject
passive voice	*based, found, shipped* since it can hide the subject
immeasurable quantification	*abundant, far, always, all, ...*
incomplete sentences	missing critical details of who must do something or what must be done

Table 2: Defects detected by QVscribe

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*, like in "The system sends an email alert to all administrators/managers"
UI details	requirements should not contain details of the user interface.
cloning	since duplicates burden successive maintenance

Table 3: Defects detected by Requirements Scout

tencies and possible similarities. In addition, it allows the generation of detailed reports that can be used to increase clarity and consistency of requirements, reducing the review and rewriting work and avoiding that critical errors can manifest themselves in the later stages of development. The defects detected by the tool and the related indicators can be classified according to Table 2.

2.3 Requirements Scout

Requirements Scout is a tool developed by Qualicen GmbH, to analyze requirements specifications (https://www.qualicen.de/en/). Requirements Scout, besides analyzing NL requirements with the aim of identifying the defects, also allows the analyst 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.

The defects and indicators of Requirements Scout are shown in Table 3.

2.4 Semantic Processing Platform

Semantic Processing Platform (Semantha) addresses the comparison of documents at the semantic level. It has been developed by thingsThinking (https://www.thingsthinking.net/). Semantha is able to search for common concepts in different documents, comparing them and highlighting the differences.

2.5 ReqSuite

The ReqSuite tool, produced by OSSENO Software GmbH (https://www.osseno.com/en/), supports rigorous requirement definition by assisting the writer who is asked to follow some patterns. For instance, requirements have to be formulated as “The system shall”, and some fields have to be filled. These fields are: ID, benefit, damage, costs, priority, type (functional/quality) and source (to specify the stakeholder requiring that requirement).

R1	The system shall enable 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.

Table 4: Requirements of the E-shop case study

2.6 A first result: tools suited to detect variation points

A preliminary analysis has shown that only two of the considered industrial tools, namely QVscribe and Requirements Scout, are suited to detect ambiguity defects, returning an output similar to QuARS.

ReqSuite performs a structural analysis, but it does not address the detection of lexical or syntactical ambiguities that can be of help to single out variation points. For instance: if a weak verb or a passive voice are used, the tool generically asks to *rephrase the sentence into a "shall" or "should" statement*; a vague term as *various* and an optional term as *possibly* are not detected.

Semantic Processing Platform could actually be used to search for variabilities, but with a different technique, namely exploiting a *contrastive analysis*. This is a known approach to apply NLP techniques to extract information related to features from existing NL documents [FSD13, FSGD15, NBA$^+$17]. The idea is that of extracting candidate terms from different documents describing a product, and identifying commonalities and variabilities to be collected in a product family, as, e.g., in [BH11]. Our technique, on the contrary, exploiting ambiguity detection, permits to extract variability from a single requirement document. Semantic Processing Platform is hence not suited to this kind of analysis since it is not designed to detect ambiguity.

As a consequence, in the following we apply only QVscribe and Requirements Scout to work out a fine grained comparison with QuARS.

3 Application of the tools to a E-shop case study: Results and Observations

QuARS, QVscribe and Requirements Scout are compared first for their general qualities, then in their ability to support a variability extraction process. The running example is a simple E-shop [AFGS20], whose requirements are presented in Table 4.

In [FFGS18b, AFGS20] we have shown that the QuARS ambiguity indicators that proved most useful to indicate variation points are multiplicity and weakness, but also optionality is another natural candidate to detect variability. The experiment presented here is aimed to understand if the other tools perform better at this regard, and to understand if they are able to reveal any fresh indicator relevant to detect variability.

3.1 General qualities comparison

We first address *documentation, learnability*, and *usability*. QuARS was simple to learn and use without referring to any manual, also by the first author who was a newcomer to our project. 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, the first author, which had no previous experience on any of the three tools, needed the following number of hours of training in order to proficiently use them: 30 for QuARS, 36 for

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 based	universal quantifiers 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 when –	optionality variability multiplicity
	QVscribe	possibly when no found –	optional escape clauses immeasurable quantification universal quantifiers passive voice no imperatives
	Req. Scout	possibly found –	vagueness 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 his	superfluous infinitives 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 and/or	vagueness dangerous slash
R7 The system shall maintain customer email information as a required part of customer profile	**QuARS**	–	multiplicity
	QVscribe	maintain as	superfluous infinitives 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**	– or available	multiplicity optionality variability
	QVscribe	shipped	passive voice
	Req. Scout	shipped	passive voice
R12 The system shall enable the user to select the shipping method	**QuARS**	– select	multiplicity variability
	QVscribe	enable	vague words
	Req. Scout	–	–
R13 The system may display the current tracking information about the order	**QuARS**	may	weakness
	QVscribe	may about –	optional escape clauses vague words no imperatives
	Req. Scout	current may –	vagueness occurrence of will or may exactly one shall or should
R14 The system shall display the available payment methods	**QuARS**	available	variability
	QVscribe	–	–
	Req. Scout	–	–
R15 The system shall allow the user to select the payment method for order	**QuARS**	select	variability
	QVscribe	allow	superfluous infinitives
	Req. Scout	–	–
R16 After delivery, the system may enable the users to enter their reviews or ratings	**QuARS**	– or may	multiplicity optionality weakness
	QVscribe	after may enable their –	non-specific temporal words optional escape clauses vague words cross-referencing pronouns no imperatives
	Req. Scout	their may –	vague pronouns 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**	needs	cross-tree constraints
	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**	excludes	cross-tree constraints
	QVscribe	–	no imperatives
	Req. Scout	–	exactly one shall or should

Table 5: Results of the application of QuARS, QVscribe, and Requirements Scout to the e-shop case study. Requirements **R5** and **R8** contain no defect according to all tools.

	QuARS			Qvscribe			Requirements Scout		
E-shop	*F.Pos.*	*Amb.*	*Var.*	*F.Pos.*	*Amb.*	*Var.*	*F.Pos.*	*Amb.*	*Var.*
Vagueness	-	-	1	4	-	-	2	-	2
Optionality	-	-	4	-	-	1	n.a.		
Weakness	-	-	2	-	-	2	-	-	2
Multiplicity	5	-	3	n.a.			n.a.		
Under-Specificaiton	-	-	-	n.a.			n.a.		
Variability	1	-	4	n.a.			n.a.		
Cross-tree Constraints	-	-	2	n.a.			n.a.		
Imperatives	n.a.			-	-	5	-	-	5
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	-	-

Table 6: Summary of variability detection (n.a. means not applicable)

QVscribe, 48 for Requirements Scout.

Efficiency was found to be an issue for Requirements Scout – to analyze documents of 20 and 50 requirements the tool takes 1 minute and 2 minutes 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 variability detection, rather than a generic low performance of the tool.

Extensibility is represented by the ability to add new quality indicators. This is permitted in QuARS, which also enables the user to select the indicators she want 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. Moreover, results can be filtered to focus on specific defects.

Other qualities that are worth mentioning 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. This feature also permits to correct the requirements on the fly and run the analysis again.

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 Comparison of the ability to detect variability

We now focus on comparing the three tools from the point of view of the ability of their indicators to detect variation points. We are interested in the quality indicators for which, using QVscribe or Requirements Scout, we have detected a different number of variabilities than using QuARS. We report in Table 5 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). These results have then been manually analysed to see whether the defects could actually be seen as variation points. The detailed results of this analysis are discussed in the following indicator by indicator. Table 6 cumulatively shows the number of *false positives, ambiguities*, and *variabilities*, with respect to the notion of variation point, as the result of the manual analysis of the tools' outcome of Table 5.

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 variability (*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 variabilities 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 variabilities 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 three variabilities related to *multiplicity* in QuARS are indeed conjunctions as in **R4** or disjunctions as in **R11, R16**. Conjunctions are not really a variability indication, they simply show that all alternatives are mandatory: in software product lines terminology, customer profile and credentials in **R4** are two *mandatory features* of E-shop. Disjunctions in **R11, R16** are also detected by *optionality* indicators.

Variability and *cross-tree constraints*, the new indicators added to QuARS precisely to detect variation points, are found in a number or requirements. There is a false positive, in **R3**, and four variabilities, in **R11, R12, R14, R15**. Requirements **R17** and **R18** contain constraints among features.

Looking at Table 6, we notice that the absence of imperatives is a main variability indicator. This is an indicator considered by QVScribe (*no imperatives)* and in Requirement Scout (*exactly one shall or should*), but not by QuARS, hence this finding is an answer to our quest for new indicators. However, we can notice that the requirements lacking an imperative (**R3, R13, R16, R17, R18**) are those containing terms such as *if possible*, or weak verbs such as *may* or *can* or cross-tree constraints. QuARS captures them with other indicators, namely *optionality, weakness* and *cross-tree constraints* indicators.

To conclude, we observe that none of the indicators going, in Table 6, from vague pronouns to UI details, that are present in QVscribe or in Requirement Scout and not in QuARS, prove useful in detecting variability.

4 Related Work and Conclusions

Ambiguity detection in requirements is a lively research field, with several contributions published already in the nineties (e.g., the ARM tool [WRH97]). Most of the works stem from the typically defective terms and constructions classified in the ambiguity paper of Berry *et al.* [BK04]. Based on these studies, rule-based NLP tools such as QuARS [GLT05], SREE [TB13] and the tool of Gleich *et al.* [GCK10] were developed. More recently, industrial applications of these approaches were studied by Femmer *et al.* [FFWE17] and by Ferrari *et al.* [FGR+18]. Furthermore, Arora *et al.* [ASBZ15] presented RETA (REquirements Template Analyzer), a tool that employs rule-based approaches to detect typical ambiguous terms and constructions, as the other mentioned works, and, in addition, checks the conformance of the requirements to a given template.

As shown also in these studies, rule-based approaches tend to produce a high number of false positive cases – i.e., linguistic ambiguities that have one single reading in practice. Hence, *statistical* approaches were proposed by Chantree *et al.* [CNDRW06] and by Yang *et al.* [YDRG+10] to reduce the number of false positive cases, referred as *innocuous ambiguities*. Statistical NLP approaches are also used in [FDG17], to identify domain-dependent ambiguities, i.e., pragmatic ambiguities that depend on the domain background of the reader of the requirements.

Our work differs from the contributions in this field, in that it integrates the research in ambiguity detection, with the research in feature identification. More specifically, we use the ambiguity detection capabilities of the considered tools to identify variation points in requirements documents. The closest works in feature identification are those that focus on variant feature identification from NL documents, as, e.g., [FSD13, NBA+17]. However, these works leverage the automated extraction of domain-specific terms, while in our work we focus on ambiguity detection.

An alternative to the experimentation of off-the-shelf tools that we have proposed in this paper is the adoption and customization of more general and flexible NLP tools, that allow to tune the kind of ambiguities and other defects that can be used as variability indicators. GATE [Cun02] 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 [FGR+18] 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. This alternative, that requires a substantial work to build a new framework on top of GATE, has not been considered in this paper, which was limited to a comparison of existing tools: anyway, the information gathered by the presented comparison can provide useful input to a future effort dedicated to build (e.g. on top of GATE) a performant customized NLP tool to extract variability information from requirement documents.

Summing up, our focus on the ambiguity indicators provided by the considered tools have soon brought us to exclude Semantha and ReqSuite as not useful to detect variability. The experimentation of the three remaining tools has shown that QuARS performs slightly better than QVscribe, followed by Requirement Scout, in terms of the number of variabilities that their indicators were able to identify. On the other hand, the latter tools have pointed at the *absence of imperatives* as a main variability indicator, not provided by QuARS, so providing an answer to our quest for new indicators. These preliminary results need to be confirmed by extending the experiments to other requirements documents; we are currently considering two publicly available larger case studies, namely *Blit* taken from [FFGS18b] and *Digital Home* taken from [FFGS18a]. Blit is a draft of the functional specication of the 55 requirements of a business project management tool, DigitalHome specifies the requirements for the development of a Smart House, where a resident can manage devices that control the environment of the home. Although we have not completed the analysis of these case studies, they seem to confirm the results achieved on the E-shop example. Our study was limited to look at the tools presented in the last edition of NLP4RE, but other tools should be considered as well. One candidate is RAT (Requirements Authoring Tool) from REUSE (https://www.reusecompany.com/), that is able to detect ambiguities, but that, for its commercial nature, was not available for our study.

Acknowledgements

We gratefully thank the developers of the tools (QVscribe, Requirements Scout, Semantic Processing Framework, and ReqSuite) that provided us with the licence for academic purposes.

The research has been partially supported by the MIUR project PRIN 2017FTXR7S "IT-MaTTerS" (Methods and Tools for Trustworthy Smart Systems).

References

[AFGS20] E. Arganese, A. Fantechi, S. Gnesi, and L. Semini. Nuts and bolts of extracting variability models from natural language requirements documents. In *Integrating Research and Practice in Software Engineering*, volume 851 of *Studies in Computational Intelligence*, pages 125–143. Springer, 2020.

[Arr19] M. Arrabito. Strumenti di analisi dei requisiti per specificare linee di prodotti, 12 2019. Bachelor thesis, Dipartimento di Informatica, Univ. di Pisa, (In italian).

[ASBZ15] C. Arora, M. Sabetzadeh, L. Briand, and F. Zimmer. Automated checking of conformance to requirements templates using natural language processing. *IEEE Trans. on Software Engineering*, 41(10):944–968, 2015.

[BH11] E. Boutkova and F. Houdek. Semi-automatic identification of features in requirement specifications. In *RE 2011, 19th IEEE Int. Requirements Engineering Conf.*, pages 313–318. IEEE, 2011.

[BK04] D.M. Berry and E. Kamsties. Ambiguity in requirements specification. In *Perspectives on software requirements*, volume 753 of *The International Series in Engineering and Computer Science*, pages 7–44. Springer US, 2004.

[CNDRW06] F. Chantree, B. Nuseibeh, A. De Roeck, and A. Willis. Identifying nocuous ambiguities in natural language requirements. In *Proceedings of the 14th IEEE International Conference on Requirements Engineering*, pages 59–68, Minneapolis/St.Paul, MN, USA, September 2006. IEEE.

[Cun02] H. Cunningham. Gate, a general architecture for text engineering. *Computers and the Humanities*, 36(2):223–254, 2002.

[DFF^{+}19] F. Dalpiaz, A. Ferrari, X. Franch, S. Gregory, F. Houdek, and C. Palomares. NLP4RE. In *Joint Proceedings of REFSQ-2019 Workshops, Essen, Germany*, CEUR Workshop Proceedings. CEUR-WS.org, 2019.

[FDG17] A. Ferrari, B. Donati, and S. Gnesi. Detecting domain-specific ambiguities: an NLP approach based on wikipedia crawling and word embeddings. In *Proc. 4th Int. Workshop on Artificial Intelligence for Requirements (AIRE), 25th Int. Requirements Engineering Conf. Workshops*, pages 393–399. IEEE, 2017.

[FFGS18a] A. Fantechi, A. Ferrari, S. Gnesi, and L. Semini. Hacking an ambiguity detection tool to extract variation points: an experience report. In *Proc. 12th Int. Workshop on Variability Modelling of Software-Intensive Systems (VAMOS), Madrid*, pages 43–50. ACM, 2018.

[FFGS18b] A. Fantechi, A. Ferrari, S. Gnesi, and L. Semini. Requirement engineering of software product lines: Extracting variability using NLP. In *Proc. 26th IEEE International Requirements Engineering Conference, Banff, AB, Canada*, pages 418–423. IEEE, 2018.

[FFWE17] H. Femmer, D.M. Fernández, S. Wagner, and S. Eder. Rapid quality assurance with requirements smells. *Journal of Systems and Software*, 123:190–213, 2017.

[FGR^{+}18] A. Ferrari, G. Gori, B. Rosadini, I. Trotta, S. Bacherini, A. Fantechi, and S. Gnesi. Detecting requirements defects with NLP patterns: an industrial experience in the railway domain. *Empirical Software Engineering*, 23(6):3684–3733, 2018.

[FGS17] A. Fantechi, S. Gnesi, and L. Semini. Ambiguity defects as variation points in requirements. In *Proc. 11th Int. Workshop on Variability Modelling of Software-intensive Systems (VAMOS)*, pages 13–19, Eindhoven, 2017. ACM.

[FSD13] A. Ferrari, G.O. Spagnolo, and F. Dell'Orletta. Mining commonalities and variabilities from natural language documents. In *Proceedings of the 17th International Software Product Line Conference, SPLC 2013, Tokyo, Japan - August 26 - 30, 2013*, pages 116–120, 2013.

[FSGD15] A. Ferrari, G.O. Spagnolo, S. Gnesi, and F. Dell'Orletta. CMT and FDE: tools to bridge the gap between natural language documents and feature diagrams. In *Proc. 19th International Conference on Software Product Line, SPLC 2015, Nashville, USA*, pages 402–410. ACM, 2015.

[GCK10] B. Gleich, O. Creighton, and L. Kof. Ambiguity detection: Towards a tool explaining ambiguity sources. In *Requirements Engineering: Foundation for Software Quality, 16th Int. Working Conference, REFSQ 2010, Essen, Germany*, volume 6182 of *LNCS*, pages 218–232. Springer, 2010.

[GLT05] S. Gnesi, G. Lami, and G. Trentanni. An automatic tool for the analysis of natural language requirements. *Computer Systems: Science & Engineering*, 20(1), 2005.

[NBA^{+}17] S. Ben Nasr, G. Bécan, M. Acher, J.B. Ferreira Filho, N. Sannier, B. Baudry, and J.-M. Davril. Automated extraction of product comparison matrices from informal product descriptions. *Journal of Systems and Software*, 124:82–103, 2017.

[PBvdL05] K. Pohl, G. Böckle, and F. van der Linden. *Software Product Line Engineering - Foundations, Principles, and Techniques.* Springer, 2005.

[TB13] S.F. Tjong and D.M. Berry. The design of SREE - a prototype potential ambiguity finder for requirements specifications and lessons learned. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, volume 7830 of *LNCS*, pages 80–95, Essen, Germany, 2013. Springer.

[WRH97] W. Wilson, L. Rosenberg, and L. Hyatt. Automated analysis of requirement specifications. In *Proc. of the 19th Int. Conf. on Software Engineering (ICSE)*, pages 161–171, Boston, 1997. ACM.

[YDRG^{+}10] H. Yang, A. De Roeck, V. Gervasi, A. Willis, and B. Nuseibeh. Extending nocuous ambiguity analysis for anaphora in natural language requirements. In *Proceedings of RE 2010, 18th International Requirements Engineering Conference*, pages 25–34, Sydney, Australia, 2010. IEEE.