

# Rule-based NLP vs ChatGPT in Ambiguity Detection, a Preliminary Study

Alessandro Fantechi<sup>1,2,\*</sup>, Stefania Gnesi<sup>2</sup> and Laura Semini<sup>2,3</sup>

<sup>1</sup>Dip. di Ingegneria dell'Informazione, Università di Firenze

<sup>2</sup>Istituto di Scienza e Tecnologie dell'Informazione "A.Faedo", Consiglio Nazionale delle Ricerche, ISTI-CNR, Pisa

<sup>3</sup>Dipartimento di Informatica, Università di Pisa

## Abstract

With the rapid advances of AI-based tools, the question of whether to use such tools or conventional rule-based tools often arises in many application domains. In this paper, we address this question when considering the issue of ambiguity in requirements documents. For this purpose, we consider GPT-3 that is the third-generation of the Generative Pretrained Transformer language model, developed by OpenAI and we compare its ambiguity detection capability with that of a publicly available rule-based NLP tool on a few example requirements documents.

## Keywords

Ambiguity detection in requirements, chatGPT, rule-based NLP tools

## 1. Introduction

GPT-3 is the third-generation of the Generative Pretrained Transformer language model, developed by OpenAI, it is an autoregressive language model and it is the largest language model constructed to date. Having sufficient data, GPT-3 can solve all kinds of tasks: it did not have any fine-tuning to solve specific tasks, like translation or text generation [1, 2]. chatGPT is a GPT-3 based conversational chatbot that has gained popularity in recent months. It is designed to respond to questions and provide information in a conversational manner, using specific training to handle conversational text and generate natural and coherent responses.

Until now, attempts to define AI-based tools for analyzing software requirements have faced the well-known lack of a corpus of annotated requirements documents on which to train the models. Some existing NLP tools harness the power of machine learning for linguistic analysis of the NL, supported by the very large size of the examples data that can be used to train the learning model, and integrate AI based language analysis with a rule-based system for ambiguity search in requirements, but they cannot be considered AI tools [3, 4].

---

In: A. Ferrari, B. Penzenstadler, I. Hadar, S. Oyedeleji, S. Abualhaija, A. Vogelsang, G. Deshpande, A. Rachmann, J. Gulden, A. Wohlgemuth, A. Hess, S. Fricker, R. Guizzardi, J. Horkoff, A. Perini, A. Susi, O. Karras, A. Moreira, F. Dalpiaz, P. Spoletini, D. Amyot. *Joint Proceedings of REFSQ-2023 Workshops, Doctoral Symposium, Posters & Tools Track, and Journal Early Feedback Track. Co-located with REFSQ 2023. Barcelona, Catalunya, Spain, April 17, 2023.*

\*Corresponding author.

✉ alessandro.fantechi@unifi.it (A. Fantechi); stefania.gnesi@isti.cnr.it (S. Gnesi); laura.semini@unipi.it (L. Semini)

↳ 0000-0002-4648-4667 (A. Fantechi); 0000-0002-0139-0421 (S. Gnesi); 0000-0001-8774-2346 (L. Semini)

 © 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

Being GPT-3 the largest language model constructed to date, we decided it was worth trying to evaluate its ability to analyze software requirements, and to compare its performance against a traditional rule-based NLP tool.

In this paper, we present a first step in this direction, in which we compared on a few requirements documents examples the ambiguity detection ability of chatGPT with that of a publicly available rule-based NLP tool, QuARS, that we already used in a previous work for ambiguity and variability detection in requirements [5, 6, 7].

The experiments described below aim at giving a first answer to the following research questions: **RQ1** Can chatGPT be used to detect ambiguities in requirements? **RQ2** How does the chatGPT performance for ambiguity detection compare to a rule based NLP tool?

The scope of the experiments is limited to four requirements documents and to a single query asked to chatGPT; however, since chatGPT returns different answers when the same question is asked again, we have run each query a few times.

Section 2 briefly introduces the issue of ambiguity detection in requirements, and the two different detection approaches of the two tools. Section 3 describes the example requirements documents used as a benchmark. The analysis of the data generated by the experiments in view of the research questions is addressed in Section 4. Final sections on threats to validity, lessons learned and conclusions follow.

## 2. Ambiguity detection

Software requirements are normally expressed informally through natural language sentences, which are potentially ambiguous, and this ambiguity is a known source of problems in the later stages of software development. In the requirement engineering community, many tools have been developed to help the analyst in detecting ambiguous requirements.

### 2.1. Rule based NLP tools for ambiguity detection

In the last decades some tools (e.g. [8, 9, 10, 11, 12, 13]) have been defined that address the automated analysis of requirements documents by means of Natural Language Processing (NLP) tools [14] with the purpose of detecting ambiguities in them. This kind of analysis is aimed at identifying typical natural language defects, especially focusing on ambiguity sources. We list in Table 1 the most common sources of ambiguity, with a classification inspired by [15, 16, 17].

As a representative of these NLP tools, in this work we apply QuARS - Quality Analyzer for Requirement Specifications, developed in our lab [18], which shows a good performance when compared with similar tools [7]. QuARS performs an automatic linguistic analysis of a requirements document in plain text format, according to the deterministic rules defined by a given quality model. Its output indicates the defective requirements and highlights the words that reveal the defect. The defect identification process includes *lexical* and *syntactical analysis*, while *semantic analysis* is not supported.

**Table 1**  
Ambiguity classes and indicators.

Ambiguity classes	Indicators
<b>Homonymy and polisemy</b>	occur when a term can have different meanings, having different (homonymy) or one (polisemy) etymology some examples are: <i>bank, can, bat... (homonyms), left, right, fall, minute, ... (polysemes)</i>
<b>Analytical, attachment, coordination</b>	occur when a sentence admits more than one grammatical structure, and different structures have different meanings syntactic analysis: the sentence admits two or more syntactic trees
<b>Anaphora</b>	occurs when an element of a sentence depends for its reference on another, antecedent, element and it is not clear to which antecedent it refers relative and demonstrative pronouns: <i>that, which, their, it, them, they, both,...</i>
<b>Vagueness</b>	occurs when it is not possible to interpret a sentence in a unequivocal way <i>clear, easy, strong, good, bad, adequate, tall, short, various, completed, similar, similarly, accordingly,...</i>
<b>Comparatives &amp; superlatives</b>	occurs when the term of comparison or the universe of discourse are missing <i>better, easier, worst, faster, bigger, biggest,...</i>
<b>Disjunctions</b>	occurs when a sentence admits different models in which the first, the second or both disjuncts are true <i>or, and/or,...</i>
<b>Escape clauses</b>	occurs when a sentence admits different models, containing or not the object the escape clause <i>case, possibly, if possible, if appropriate, among others, as a minimum, when required, ...</i>
<b>Weakness</b>	occurs when the sentence contains weak verbs <i>may, can, could,...</i>
<b>Quantifiers</b>	in presence of quantifiers, ambiguities are due to the scope or to the universe of quantification <i>a, all, always, every, any, nothing,...</i>
<b>Under-specification</b>	occurs when the sentence contains terms that need to be instantiated or qualified <i>information, interface, attack, button, channel, component, procedure, process, report, session,...</i>
<b>Passive voice</b>	occurs when the subject of the passive sentence is not be revealed auxiliary <i>to be</i> with a past participle and no agent specified ( <i>by</i> )

## 2.2. chatGPT for ambiguity detection

As an AI large language model (LLM), chatGPT doesn't use rules to detect ambiguities in the traditional sense. Instead, it uses training data and algorithms to generate an answer. LLMs are such complex algorithms that it is arduous, if not infeasible, to know exactly how and why the model returns a particular result (lack of explainability and transparency) and it is rare to get the same answer twice (nonreproducibility). These are well-known issues that need to be considered when switching from rule-based approaches to LLMs, particularly if there is a need to guarantee a quality level of the requirements. The purpose of this work, however, is to investigate whether chatGPT has reasonable performance in ambiguity detection compared with rule-based tools, such that it would make it a useful tool in software development, alone or in combination with rule-based tools. To the best of our knowledge, there is no documentation or literature so far on the ambiguity detection capabilities of chatGPT.

## 3. Data preparation

To perform our experience we have used two simple requirements documents introduced in previous papers, and two third-party requirements documents<sup>1</sup>:

**Coffee machine** that gives few requirements of an automatic coffee vending machine;

---

<sup>1</sup>All documents are available at <https://github.com/Vibe-NLP/RequirementsForValidation>.

**Table 2**

Characteristics of the requirement documents: number of requirements, number of words, authorship and characteristic of the system to be.

	reqs	words	issued by	characteristics
Coffee machine	6	63	authors	toy example
E-shop	18	263	authors	toy example
Library	94	1815	company	information system
DigitalHome	112	1121	academia	control system

**Table 3**

Coffee-machine requirements

- 
- C1 After inserting a suitable coin, the user shall choose a beverage and select the amount of sugar.
  - C2 The machine shall offer, as beverages, Coffee and Cappuccino or Tea.
  - C3 The machine shall always offer coffee.
  - C4 A ringtone possibly has to be played after beverage delivery.
  - C5 After the beverage is taken, the machine returns idle.
  - C6 The British market requires tea and excludes any ring tone.
- 

**Table 4**

E-shop requirements

- 
- E1 The system shall enable the user to enter the search text on the screen.
  - E2 The system shall display all the matching products based on the search.
  - E3 The system possibly notifies with a pop-up the user when no matching product is found on the search.
  - E4 The system shall allow a user to create his profile and set his credentials.
  - E5 The system shall authenticate user credentials to enter the profile.
  - E6 The system shall display the list of active orders and/or the list of completed orders in the customer profile.
  - E7 The system shall maintain customer email information as a required part of customer profile.
  - E8 The system shall send an order confirmation to the user through email.
  - E9 The system shall allow an user to add and remove products in the shopping cart.
  - E10 The system shall display various shipping methods.
  - E11 The order shall be shipped to the client address or, if the shipping to store service is available, to an associated store.
  - E12 The system shall enable the user to select the shipping method.
  - E13 The system may display the current tracking information about the order.
  - E14 The system shall display the available payment methods.
  - E15 The system shall allow the user to select the payment method for order.
  - E16 After delivery, the system may enable the users to enter their reviews and ratings.
  - E17 Shipping time should be as fast as possible.
  - E18 The system must report the available products, if the availability of these are less than 10 percent the system should show a pop-up.
- 

**E-shop** that describes a simple online shopping system;

**Library**, that describes the requirements for the System Administration Module of a urban library system.

**DigitalHome**, that specifies the requirements for developing a domotic system.

In Table 2 we summarise some characteristics of the considered documents. In Tables 3 and 4 we present the requirements of the coffee machine and E-shop, respectively.

## 4. Data Collection and Analysis

To address the RQs, including RQ2 that requires a comparison with a rule based NLP tool, we perform the following steps:

**Automatic detection:** We apply both QuARS and chatGPT to each document. The document is given as input to QuARS in text format while chatGPT is queried by asking: "Find the ambiguities of the following software requirements document: *<list of requirements in text format>*".

QuARS returns the requirements that are considered ambiguous, along with the term or expression that is an indicator of ambiguity and the defect class to which it refers. chatGPT has a less structured and more variable response format, but basically indicates which requirements are ambiguous and why.

**Review:** The output of the tools is reviewed by the authors in a joint meeting and each defect identified as ambiguity or false positive. The classification derived at this stage is the one used for data analysis in the following step.

**Assessment:** The analysis is both quantitative, in terms of performance metrics, and qualitative, to understand in detail what kind of defects are identified or ignored by the two tools.

For the quantitative analysis, we use the following metrics, where  $tp$  is true positive,  $fp$  is false positive and  $fn$  is false negative:

$$\text{precision} = \frac{tp}{tp + fp} = \frac{|\text{found ambiguities} \cap \text{true ambiguities}|}{|\text{found ambiguities}|}$$
$$\text{recall} = \frac{tp}{tp + fn} = \frac{|\text{found ambiguities} \cap \text{true ambiguities}|}{|\text{true ambiguities}|}$$

### 4.1. Coffee machine

As an initial case study we consider the coffee machine, which is short enough to allow us to fully report the output of QuARS and the response to a query to chatGPT. We are well aware that considering only one response is statistically irrelevant, but this example is mainly for illustrative purposes. Let us first report the output of QuARS:

The line number: 1. after inserting a suitable coin, the user shall choose a beverage and select the amount of sugar.

is defective because it contains the wording: **suitable**

The line number: 2. the machine shall offer, as beverages, coffee and cappuccino or tea. is defective because it contains the wording: **or**

The line number: 3. the machine shall always offer coffee.  
is defective because it contains the wording: **always**

The line number: 4. a ringtone possibly has to be played after beverage delivery.  
is defective because it contains the wording: **possibly**

The line number: 6. the british market requires tea and excludes any ring tone.  
is defective because it contains the wording: **any**

Then we have asked chatGPT obtaining the following answer:

**Table 5**

Coffee machine case study. We report the indicator found with its defect class.

Req	QuaRS		chatGPT		Manual analysis Indicator
	Indicator	Defect	Indicator	Defect	
C1	suitable	vagueness	suitable	vagueness	suitable
C2	or	disjunction	or	ambiguous disjunction	or coord. ambiguity
C3	-	-	-	-	-
C4	always	quantification	always	contradiction	-
C5	possibly	optionality	possibly	optionality	possibly passive voice
C6	-	-	-	-	-
C7	any	quantification	-	incompleteness	-

1. It is not clear what is meant by "a suitable coin." Is there a specific type of coin that the machine will accept, or will it accept any coin?
2. The statement "The machine shall offer, as beverages, Coffee and Cappuccino or Tea" is ambiguous because it is not clear whether the machine will offer all three options (Coffee, Cappuccino, and Tea) or only one of the options (Coffee, Cappuccino, or Tea).
3. The statement "The machine shall always offer coffee" seems to contradict the previous statement about the available beverage options.
4. It is not clear what is meant by "a ringtone possibly has to be played after beverage delivery." Is this a requirement or just a possibility?
5. The statement "The British market requires tea and excludes any ring tone" does not specify how the machine should handle these requirements. Is the machine supposed to automatically adjust its behavior for the British market, or is this something that the user needs to manually set?

## 4.2. Results analysis and comparison

We have manually analysed the requirements, according the classification of ambiguity sources in Table 1 and then assessed the tools' outcome. Results are summarised in Table 5 and commented below; performance results are non-significant in this toy example.

1. "suitable coin" in C1 is an ambiguity, detected by both tools;
2. "or" in C2 is an ambiguity detected by both tools. In the same requirement there is a coordination ambiguity, undetected by the tools;
3. "always" in C3 is a false positive, detected as ambiguity by QuaRS. ChatGPT returns an indication of a possible contradiction, which might exist, but is not an ambiguity;
4. the fact that the ring tone is *possibly* played, in C4, is an ambiguity and it is detected by both tools;
5. in C6 QuaRS finds "any", which is a false positive, while chatGPT detects an incompleteness that actually exists, but is not an ambiguity.

## 4.3. E-shop

Our second experience involved the E-shop example: we performed a manual analysis, an analysis with QuaRS, and queried chatGPT twice, on different days. For space reasons, we do not report the whole outcomes but only the found indicators and kind of defect in Table 6. Performance values are in Table 7 and show that the performance of chatGPT can be highly

**Table 6**

E-shop case study. All indicators found are true positives unless labeled as false positives (fp).

Req	QuaRS		chatGPT 1		chatGPT 2		Manual analysis
	Indicator	Defect	Indicator	Defect	Indicator	Defect	
E2	all	quantif. (fp)	-	-	-	-	-
E3	possibly	optionality	possibly	optionality	-	-	possibly
-	-	-	-	-	no	quantif.	-
E6	and/or	optionality	and/or	ambig. disj.	and/or	ambig. disj.	and/or
E10	various	vagueness	-	-	-	-	various
E11	or	optionality	or	ambig. disj.	-	-	or
-	-	-	-	-	associated	vague (fp)	-
E13	may	weakness	may	weakness	-	-	may
-	-	-	-	-	current	unclear (fp)	-
E16	may	weakness	may	weakness	-	-	may
E17	should	weakness	should	replace by shall	-	-	should as fast as possible
-	-	-	-	-	as fast as possible	subjective	
E18	should	weakness	should	replace by shall	-	-	should

**Table 7**

E-shop and Smart Home: performance measures.

	QuaRS		chatGPT1		chatGPT2	
	precision	recall	precision	recall	precision	recall
eshop	0,89 (8/9)	0,8 0,89 (8/9)	1 (7/7)	0,78 (7/9)	0,6 (3/5)	0,33 (3/9)
smart_home	0,24 (17/70)	0,77 (17/22)	0,28 (3/14)	0,14 (3/22)	0,17 (2/12)	0,09 (2/22)

variable, which was expected, but also that it can be compared with that of a settled, rule-based tool. It is interesting to note that chatGPT was able to detect an hidden ambiguity in E3 that was not found by manual analysis (note also that we have been working on for some time on this case study and we had never noticed the problem):

Ambiguity in E3: It is not specified if the pop-up will be displayed when a single product is not found or multiple products.

#### 4.4. Smart Home

Our third experience involved the smart home example: we performed an analysis with QuARS, and queried chatGPT as usual. We encountered a limitation of chatGPT: it does not accept documents of this length as input and returns an error. To get around the problem, we divided the document into two parts and had them analysed separately, then merged the results. Since we are looking for sources of ambiguity, which do not depend on the joint analysis of multiple requirements, we claim that this partition-based solution is acceptable. Each part has been analysed twice, on different days.

This document returned many false positives, both with QuARS and chatGPT, and chatGPT has a very low recall (Table 7). For space reasons, we do not present the tools output but only the performance measures. With regard to qualitative analysis, chatGPT found the following defect not found by QuARS, which is worth noting since it reveals an incompleteness of the QuARS dictionaries, which do not contain the term *compatible*.

Ambiguity in 2.2.5, as it states "The system shall be compatible with a centralized HVAC

**Table 8**

Library: performance measures.

Library	QuaRS		chatGPT 1 <sup>st</sup> run		chatGPT 2 <sup>nd</sup> run		
	precision	recall	precision	recall	precision	recall	
	0,46 (12/26)	0,55 (12/22)	0,5 (2/4)	0,09 (2/22)	0,75(3/4)	0,04 (1/22)	
chatGPT 3 <sup>rd</sup> run		chatGPT 4 <sup>th</sup> run		chatGPT 5 <sup>th</sup> run			
precision	recall	precision	recall	precision	recall		
	0,5 (4/8)	0,18 (4/22)	0,41 (9/22)	0,41 (9/22)	0,59 (10/17)	0,46 (10/22)	

(Heating, Ventilation and Air Conditioning) systems: gas, oil, electricity, solar, or a combination of two or more." It is unclear what "compatible with" means in this context and how it applies to the different types of HVAC systems mentioned.

#### 4.5. Library

The last document considered is Library, which is slightly smaller in size than Smart Home. We analysed the document with QuaRS and then with chatGPT for 5 times, on 5 different days. In the table named GPT\_QuARS\_library in <https://github.com/Vibe-NLP/RequirementsForValidation> we list all the defects found. The table is truncated because all 5 times we queried chatGPT, although it did not report length errors, it only found defects in the first 38 requirements. We therefore decided to consider this document fragment to make the performance measurements, which are shown in Table 8. In the GPT\_QuARS\_library table, for each analysis, we show each defect reported, labelling it directly as false positive (fp) or true positive (amb). In the adjacent column we report: for QuaRS, which indicators were considered false positives or true ambiguities; for chatGPT a fragment of the response, if significant.

#### 4.6. Threats to validity

We have used precision and recall as metrics to compare the tools. The human intervention in the review and assessment steps, returning the number of true/false positives and false negatives, is a threat to *construct validity*, and the involvement of the authors in these phases is also a threat to *internal validity*. With regard to *external validity*, we have presented a preliminary study, and the quantitative comparison is limited to three case studies, to two compared tools and to a single kind of query to chatGPT and few chat sessions.

### 5. Conclusions and Future Work

The findings from the experience allow us to give an answer, albeit preliminar, to the RQs:

**RQ1** chatGPT can be used to detect ambiguities in requirements by simply asking: "Find the ambiguities of the following software requirements document:<*list of requirements in text format*>". We note that chatGPT does not process long requirement documents: either it returns an error or it provides a partial answer. Since ambiguity detection does not

depend on processing the document as a whole, it is possible to break the requirements document into simpler parts and analyze the pieces separately.

**RQ2** ChatGPT's performance results vary between chat sessions with the bot, especially recall; precision, on the other hand, is more stable and comparable to that of a rule-based NLP tool. Running several sessions with the same question improves recall. For example, when making the union of the 5 responses got from the chatbot for the library case study, we have the following performance: precision = 0,51(28/55) recall=0,55(12/22)

Validity threats can be mitigated in future work by involving third-party reviewers and measuring the level of agreement between them and by increasing the number of documents and querying chatGPT with different queries.

Future work can further develop the analysis presented here along several dimensions:

- Assess the coverage by GPT-3 language model of the technical slang used in requirements;
- Exploit ChatGPT's ability to rationalise and explain ambiguity;
- Ask ChatGPT more focused questions, addressing the various classes of ambiguity separately;
- Develop the analysis with additional documents and evaluate the hypothesis that slicing a requirements document for chatGPT does not influence its results;
- We have seen that chatGPT is able to detect defects, such as incompleteness and inconsistency, that traditional NLP tools cannot identify or can identify with difficulty and after domain-focused training. A future study may be devoted to specifically measuring the performance of chatGPT in finding these classes of defects in requirements. Positive results in this respect could lead to the use of chatGPT to complement a rule-based tool to automatically detect these important quality criteria;

## Acknowledgments

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

## References

- [1] T. B. Brown, B. Mann, et al., Language models are few-shot learners, in: 33rd Annual Conference on Neural Information Processing Systems 2020, Dec. 6-12, 2020.
- [2] R. Dale, GPT-3: what's it good for?, Nat. Lang. Eng. 27 (2021) 113–118.
- [3] IBM, Engineering Requirements Quality Assistant (RQA), . URL: [www.ibm.com/products/requirements-quality-assistant](http://www.ibm.com/products/requirements-quality-assistant).
- [4] A. Fantechi, S. Gnesi, S. Livi, L. Semini, A spaCy-based tool for extracting variability from NL requirements, in: M. Mousavi, P. Schobbens (Eds.), SPLC '21: 25th ACM Int. Systems and Software Product Line Conference, Leicester, UK, Sept. 6-11, ACM, 2021, pp. 32–35.
- [5] A. Fantechi, A. Ferrari, S. Gnesi, L. Semini, Requirement Engineering of Software Product Lines: Extracting Variability Using NLP, in: 26th IEEE International Requirements Engineering Conference 2018, Banff, Canada, August 20-24, 2018, IEEE, 2018, pp. 418–423.

- [6] A. Fantechi, S. Gnesi, L. Semini, VIBE: looking for variability in ambiguous requirements, *J. Syst. Softw.* 195 (2023).
- [7] M. Arrabito, A. Fantechi, S. Gnesi, L. Semini, An experience with the application of three NLP tools for the analysis of natural language requirements, in: Proc. of Quality of Information and Communications Technology - 13th Int. Conference, QUATIC, volume 1266 of *Communications in Computer and Information Science*, Springer, 2020, pp. 488–498.
- [8] J. Kasser, TIGER-PRO , . URL: [www.therightrequirement.com](http://www.therightrequirement.com).
- [9] V. Ambriola, V. Gervasi, Processing natural language requirements, in: Int. Conference on Automated Software Engineering, ASE, Nov. 2-5, IEEE Computer Society, 1997, pp. 36–45.
- [10] O. Kenney, M. Cooper, Automating requirement quality standards with QVscribe, in: NLP4RE'20, co-located with the 26th Int. Conf. on Requirements Engineering: Foundation for Software Quality (REFSQ), volume 2584 of *CEUR Workshop Proc.*, CEUR-WS.org, 2020.
- [11] H. Femmer, Requirements quality defect detection with the Qualicen requirements scout, in: NLP4RE'18, co-located with the 23rd Int. Conf. on Requirements Engineering: Foundation for Software Quality (REFSQ), volume 2075 of *CEUR Workshop Proceedings*, 2018.
- [12] S. F. Tjong, 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*, Springer, Essen, Germany, 2013, pp. 80–95.
- [13] R. Company, RAT, . URL: [www.reusecompany.com/rat-authoring-tools](http://www.reusecompany.com/rat-authoring-tools).
- [14] G. G. Chowdhury, Natural language processing, *Annu. Rev. Inf. Sci. Technol.* 37 (2003) 51–89.
- [15] D. Berry, E. Kamsties, M. Krieger, From contract drafting to software specification: Linguistic sources of ambiguity - a handbook version 1.0 (2003).
- [16] V. Gervasi, A. Ferrari, D. Zowghi, P. Spoletini, Ambiguity in requirements engineering: Towards a unifying framework, in: From Software Engineering to Formal Methods and Tools, and Back - Essays Dedicated to Stefania Gnesi on the Occasion of Her 65th Birthday, volume 11865 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 191–210.
- [17] INCOSE, Guide for Writing Requirements, TechGuideWR2019Soft V3, 2019.
- [18] S. Gnesi, G. Lami, G. Trentanni, An automatic tool for the analysis of natural language requirements, *Computer Systems: Science & Engineering* 20 (2005).