QuOD: an NLP tool to Improve the Quality of Business Process Descriptions

Alessio Ferrari[1 [0000−0002−0636−5663]], Giorgio O. Spagnolo[1 [1111−2222−3333−4444]], Antonella Fiscella[2], and Guido Parente[2]

[1] ISTI-CNR, Pisa, Italy
alessio.ferrari@isti.cnr.it, spagnolo@isti.cnr.it
[2] Narwhal Software, Florence, Italy
info@narwhal.it

Abstract. [**Context and Motivation**] In real-world organisations, business processes (BPs) are often described by means of natural language (NL) documents. Indeed, although semi-formal graphical notations exist to model BPs, most of the legacy process knowledge—when not *tacit*—is still conveyed through textual procedures or operational manuals, in which the BPs are specified. This is particularly true for public administrations (PAs), in which a large variety of BPs exist (e.g., definition of tenders, front-desk support) that have to be understood and put into practice by civil servants. [**Question/problem**] Incorrect understanding of the BP descriptions in PAs may cause delays in the delivery of services to citizens, or, in some cases, incorrect execution of the BPs. [**Principal idea/results**] In this paper, we present the development of an NLP-based tool named QuOD (Quality Analyser for Official Documents), oriented to detect linguistic defects in BP descriptions and to provide recommendations for improvements. [**Contribution**] QuOD is the first tool that addresses the problem of identifying NL defects in BP descriptions of PAs. The tool is available online at http://narwhal.it/quod/index.html.

Keywords: NLP · Business Process · Requirements Engineering.

1 Introduction

Public Administrations (PAs) are socio-technical systems whose goal is to provide services to citizens in accordance with the law. Services are performed by civil servants following business processes (BPs), which are sequences of activities to be carried out to deliver a service [5]. In PA, as in other organisations, BP specifications are available in the form of written procedures, or operational manuals [16, 15, 22]. As typical also for system/software requirements specifications, these documents are expressed in informal natural language, which is inherently open to different interpretations [20, 23, 2]. Hence, the content of these documents might be incorrectly interpreted by those who have to put the process into practice. It is therefore important to identify linguistic defects in written BP specifications, to ensure that BPs are properly carried out [19, 23, 12].

In the context of the EU Project Learn PAd (`http://www.learnpad.eu`) [9, 8, 10], we developed a tool, named QuOD (Quality Analyser for Official Documents), which is specifically oriented to identify language defects in written BP specifications and official documents of PAs. The tool is based on the evaluation of a set of quality attributes, with associated indicators of potential defects. Specifically, QuOD deals with four main quality attributes, namely *simplicity*, *non-ambiguity*, *content clarity* and *correctness*, and identifies defects such as the usage of difficult jargon, syntactic ambiguities, unclear actors or acronyms as well as grammatical errors. To this end, QuOD leverages a set of patterns expressed by means of the JAPE grammar, supported by the GATE (General Architecture for Text Engineering)[3] tool.

In this paper, we present the quality model developed within the context of Learn PAd, which was used as a reference to define the defect detection patterns of QuOD. Furthermore, we describe each pattern in details and we present the web interface of the tool. Further information about the development of the tool, and the role of its patterns in the context of Learn PAd can be found in our public deliverable [15].

The remainder of the paper is structured as follows. In Sect. 2 we briefly present background on the Learn PAd project and the quality model. In Sect. 4 to 6 we present the patterns associated to each quality attribute of the quality model. Sect. 7 presents the interface of QuOD, and Sect. 8 concludes the paper.

2 The Learn PAd Quality Model

The Learn PAd EU project [9, 8, 10] aims to improve the sharing of knowledge among civil servants, and as a consequence the perceived quality of services delivered by the public administration (PA). The overall idea of Learn PAd is to use the business process modeling notation (BPMN) [4] to teach civil servants how the procedures shall be implemented in practice and to complement the models expressed according to the BPMN with BP descriptions that give details in natural language about the procedures.

In the context of the project, a quality model was defined comprising a set of defects to be automatically identified in the BP descriptions. The quality model is based on a throughout domain analysis published in a recent work [16], and focuses on those defects that can be automatically checked by means of a rule-based system, i.e., a system that is based on pattern matching algorithms.

A quality model is a reference model against which a certain artifact—a PA procedure expressed in natural language, i.e., a BP description, in our case—can be evaluated [17]. A quality model is defined by means of a set of quality attributes, which are high-level quality properties that the PA procedure shall exhibit. The general quality model for PA procedures comprises seven general quality attributes, namely:

[3] `https://gate.ac.uk`

- **Clarity:** this attribute indicates that the PA procedure is understandable, both in terms of *content*, in terms of presentation, and in terms of practical applicability.
- **Non-ambiguity:** this attribute indicates that the content of the PA procedure has only one interpretation, independently of the reader. The attribute considers the non-ambiguity of terms, and the non-ambiguity of the syntax used in the sentences of the PA procedure.
- **Simplicity:** this attribute indicates that the content of a PA procedure is easy to read. The attribute considers both the difficulty of the terms and the difficulty of the syntax.
- **Completeness:** this attribute indicates that all the required fields of a given template for PA procedures are filled with content. The attribute requires a reference template to be defined.
- **Conciseness:** this attribute indicates that the PA procedure is sufficiently synthetic, and does not have any irrelevant detail or repetition.
- **Correctness:** this attribute indicates that the content of the PA procedure is correct in terms of grammar, and does not include copy-paste errors.
- **Coherence:** this attribute indicates that the content of the PA procedure is not contradictory or illogical. The attribute takes into account the internal coherence, the external coherence (i.e., the coherence with other documents), and the coherence with respect to the real world (referred as applicability incoherence).

Among the different quality attributes, in this paper we focus on those that have been addressed with the definition of a set of patterns implemented in QuOD. Specifically, we focus on (content) clarity, non-ambiguity, simplicity, and correctness. The other quality attributes can be enforced by means of the guidelines for writing BP descriptions collected by Ferrari et al. [16] and the BP description template presented therein. For each quality attribute, we have identified a set of *indicators*, which can be automatically detected and provide information about a particular attribute [17]. Indicators can be regarded as *defects* to be matched by means of defect detection patterns. Patterns are regular expressions that might involve characters or more complex linguistic constructs, such as words, and phrases. To express simple patterns we generally use an intuitive semi-formal notation that use natural language and symbols. To express more complex patterns we use a notation inspired to the JAPE grammar [27], which is the one employed by the tool GATE and that is used to implement the patterns in QuOD. Each pattern has been designed to identify the majority of potential defects. The idea, borrowed from the requirements engineering domain [1], is that the system raises the possibility of a defect in the text, and that the user considers whether such defect is an actual defect, or can be ignored. The rationale here is that a user can easily discard those potential defects that are not actual flaws from their point of view, while more severe consequences can be expected (e.g., procedure not correctly performed or not performed at all [22]) in case a defect is not detected. Each of the following sections is dedicated to a quality attribute, and to the associated indicators.

3 Quality Attribute: Non-Ambiguity

The *non-ambiguity* quality attribute defines the degree of non-ambiguity of a BP description. Such quality attribute considers both the ambiguity of the terms and the ambiguity of the syntax. The following sections describe the indicators that we consider for this attribute.

3.1 Indicator: Lexical Ambiguity

In general, a lexical ambiguity occurs whenever a term can have different meaning (e.g., the word "bank" can be the bank of a river, or the bank as "establishment for custody, loan, exchange, or issue of money") [2]. However, in this context, we will not refer to this definition of lexical ambiguity – cases as the one exemplified will be treated as *pragmatic* ambiguity, since the interpretation of "bank" depends on the context. Instead, we will refer to the model defined by Gnesi *et al.* [17], for checking the quality of natural language requirements specification. According to such model, lexical ambiguity occurs whenever a sentence includes an adverb, adjective or conjunction, possibly combined with prepositions, that might lead to different interpretations of the sentence. In practice, the considered model does not take into account *names* or *verbs* with potentially different interpretations, but solely typical expressions that are commonly source of potential misunderstandings. Four categories of lexical ambiguity are defined in [17], namely vagueness, subjectivity, optionality and weakness. The first category includes the usage of vague expressions, with a non uniquely quantifiable meaning, such as "accurate", "suitable", "appropriate", "clearly", etc. The second category includes expressions that refers to personal opinions or feelings, such as "better", "accordingly", "depending on", etc. The third category includes expressions that reveal the presence of an optional part in the sentence, such as "if necessary", "if needed", "and/or". The fourth category include cases when a weak main verb, such as "can", "may", etc., is used. Examples for the first three categories are provided below:

- **Vagueness:** *The field office will forward the application to the* ***appropriate*** *official for a final decision.* Here, the term "appropriate" is vague, and the editor shall specify which is the specific official that is in charge of taking the final decision.
- **Subjectivity:** *Support staff may be called in from other teams* ***depending on*** *the extent of the scene.* Here, the expression "depending on" leaves the reader with the freedom to personally evaluate the extent of the scene.
- **Optionality:** *The director of the group must transfer 10% of the funded loans to the institute* ***and/or*** *to the department.* Here the expression "and/or" leaves the freedom of sending the funded loans to just one organisation.

In the context of Learn PAd, we do not consider cases of of "weakness", since this indicator was specifically designed for natural language requirements specifications, and appeared less suitable for PA documents. Indeed, in the context

of PA procedure descriptions, we have found that it is rather frequent to find verbs such as "can" or "may" (e.g., 63 cases of"can", and 124 cases of "may" are found in our dataset [15]), and these are normally acceptable (as, e.g., in the following example *"Ensure you can meet the deadlines"*).

To check the presence of vagueness, subjectivity or optionality in a sentence, we define three patterns. Let V, U and O be sets of vague, subjective, or optional terms. Let S be a sentence, and let $T(S)$ be any sequence of words in the sentence. The patterns are the following:

- **VAG:** $\forall v \in V, \forall t \in T(S)$, if $t = v$, mark t as vague.
- **SUB:** $\forall u \in U, \forall t \in T(S)$, if $t = u$, mark t as subjective.
- **OPT:** $\forall o \in O, \forall t \in T(S)$, if $t = o$, mark t as optional.

If a sentence has at least one term that is detected to be vague, subjective of optional according to the at least one of the previous patterns, such sentence is marked as defective. In QuOD, we employ the dictionaries used by QuARS [17], to check the three categories of lexical ambiguity exemplified above. Therefore, the sets V (446 terms), S (19 terms) and O (11 terms) are composed of all the terms used by QuARS.

3.2 Indicator: Syntactic Ambiguity

Syntactic ambiguity manifest itself whenever the sentence can have more than one grammatical structure, each one with a different meaning. Four types of syntactic ambiguity are defined in the literature [2], namely *analytical* (i.e., a complex noun group with modifiers [18]), *attachment* (i.e., a prepositional phrase can be attached to two parts of the sentence), *coordination* (i.e., when more than one conjunction "or", or "and" is used in a sentence), *elliptical* (i.e., when words are omitted because they are expected to be deduced from the context), and *anaphoric/referential* (i.e., when pronouns or other words refer to other elements, but there is more than one possibility). This latter type of ambiguity may involve different sentences, and the literature often categorise it as pragmatic ambiguity. However, given its strong relation with the syntax, and its similarity with, e.g., attachment ambiguity, we consider more reasonable to include it among the syntactic ambiguities.

Examples of each category are provided below:

- **Analytical:** *The Italian office director.* Here, "Italian" can be referred to the office or to the director.
- **Attachment:** *The officer edits a resumee with a template for the final assessment.* Here "for" can be referred to the "template", or to the "resumee" or can specify a deadline (i.e., before the final assessment).
- **Coordination:** *The employee met the council* ***and*** *the head of office* ***and*** *the secretary assessed his presence.* Here, the sentence can have several parses. For example, it is unclear whether both the head of office and the secretary assessed the presence of the employee, or just the secretary.

- **Elliptical:** *The successful candidate receives the letter on Sept. 12, and the unsuccessful doesn't.* Here, the ambiguity is whether the unsuccessful candidate receives a notification in another date, or does not receive any notification.
- **Anaphoric:** *The delegate assesses the presence of the candidate, and* ***he*** *provides his signature.* Here "he" can be referred to both the delegate or the candidate.

We decided to focus on a sub-set of the syntactic ambiguity categories and to provide pattern-based approaches for them. The chosen categories are coordination and anaphoric ambiguities. The choice has fallen on these categories since they are more clearly defined in the literature, and can be in principle associated to the presence of specific keywords (e.g., "and", "or" for coordination ambiguities, and pronouns for anaphoric ambiguities). The other types of syntactic ambiguities are more likely to be identifiable with machine learning approaches.

Coordination Ambiguities Potential coordination ambiguities may occur when we have more than one coordinating conjunction in the form "or" or "and" in the same sentence, as in the example provided above. Moreover, they may occur when a conjunction is used with a modifier, as e.g., in the sentence *"**Novel employees and directors** are required to provide summaries of their work at the end of the year"* (is "novel" referred to employees only, or to both employees and directors?). To detect these types of ambiguity, two patterns, one for each type, can be provided.

- **CAMB-1:** (Token)* (and | or) (Token.kind != "punct")* (and | or) (Token)∗
- **CAMB-2:** (JJ) (NN | NNS) (and | or) (NN | NNS).

The first pattern searches for at least two occurrences of "and" or "or", not separated by punctuation (e.g., commas, semicolons, separator such as "-", etc.). As reported in [2], commas, and other types of punctuation may clarify the syntactic structure. Coordination ambiguity may occur also in presence of punctuation. However, we have evaluated these cases are sufficiently rare to be negligible. The second pattern matches cases where an adjective (JJ) precedes a couple of singular (NN) or plural nouns (NNS), joined by "and" or "or".

Anaphoric Ambiguities Anaphora occurs in a text whenever a linguistic expression (e.g., personal pronouns such as "he/she/it", possessive pronouns as "her/his", relative pronouns such as "that", "which", demonstrative pronouns such as "this", "who", etc.) refer to a previous part of the text. The referred part of the text is normally called *antecedent*. An anaphoric ambiguity occurs if the text offers one or more antecedent options, either in the same sentence or in previous sentences [28]. Here, we focus on anaphoric ambiguities that involve third personal subject/object pronouns and possessive pronouns, of the three genders, namely male ("he","his", "him", "himself"), female ("she", "her", "hers",

"herself"), and neuter ("it", "its", "itself", "they", "their", "theirs", "them", "themselves"). We do not focus on first and second person pronouns, since these are less frequent in PA documents.

The potential antecedents for these pronouns are noun phrases (NP) [28]. Therefore, we define the following two patterns to identify potential cases of anaphoric ambiguities.

- **AAMB-1:** (NounChunk) (NounChunk)+ (Pronoun)
- **AAMB-2:** (NounChunk) (NounChunk)+ (Split) (Pronoun)

The first pattern matches any single sentence with a pronoun and two or more potential antecedents. The second pattern searches for potential antecedents in the previous sentence (the notation "Split" indicates the sentence separator).

4 Quality Attribute: Simplicity

The *simplicity* quality attribute defines how easy is to read a BP description. It is a quality attribute that, in a sense, shall give an overall degree of readability of each sentence, and compute an aggregate value of readability. Such quality attribute takes into account the difficulty of the terms. The difficulty associated to the syntax – a topic that is still a matter of research, see. e.g., [11] – instead is considered by simply evaluating the length of the sentences. We use the term "simplicity" and not "readability", since readability in the literature is a more domain-generic concept, which involves also typographical aspects and degree of interest that a text raises [16]. Here, we wish to highlight that the defects that we address are those that makes *difficult* the understanding of PA procedure descriptions, such as, e.g., juridical jargon and difficult jargon. Therefore, we have considered the term simplicity to be more appropriate. The following sections describe the indicators that we consider for this attribute.

4.1 Indicator: Excessive length

This indicator indicates that a sentence is too long. The length of a sentence is a rather intuitive indicator of its complexity. Normally a long sentence includes multiple concepts that have to be processes by the reader, and is more likely to include complex syntactic constructions that require higher reading effort. An example of long sentence is provided below:

- **Long Sentence:** *Further distribution of vote sheets within the staff is permissible upon issuance of the vote, but distribution outside the agency is permissible only after the final collegial decision is recorded by the Secretary in an SRM to the action office and the votes have been released to the public.* This sentence is 49 words, and 293 characters, and it requires multiple readings to be understood.

This indicator can be easily checked with this basic pattern:

– **LEN:** N = number of words in a sentence, $N < \tau$.

The *The Plain English Guide* by Martin Cutts [6] states that sentences should be 15-20 words in average, and should not exceed 40 words. Moreover, the style guidelines of the English government [26] recommends sentences to not exceed 25 words. Therefore, in the context of Learn PAd, we take the threshold τ of 26 words as basic rule to check whether a sentence is too long.

4.2 Indicator: Juridical jargon

Juridical jargon is the usage of terms and constructions that belong to the juridical domain. This domain has defined a specific jargon that is understood by domain experts, and in a sense, is oriented to establish clear concepts and to avoid ambiguity. Nevertheless, studies as [25] have shown that even technical experts prefer text that use plain English instead of legal jargon, and that the more specialist the knowledge of the reader, the higher the preference for plain English. These studies have been used also by the UK government to define their guidelines for editing the content of their Web pages [26], where they recommend to minimize the usage of juridical jargon, and latin terms, which are typical in legal writing. Moreover, our interviews and questionnaires show that the presence of juridical jargon is one of the main linguistic problems found in their current procedure descriptions.

To address this problem, we define the current indicator – i.e., juridical jargon – which aims to identify juridical words and expressions in the Learn PAd content. It is worth mentioning that the term "jargon" includes not only words and expressions, but also the syntax. Here, we focus solely on the terms (i.e, words and expressions), since other indicators are defined in Learn PAd that address problem with ambiguous syntax (see Sect. 3.2), a typical problem of juridical jargon.

Let J be a set of juridical terms, let S be a sentence and let $T(S)$ be the set of any ordered sequence of words in a sentence (i.e., any potential single or multi-word term). The following pattern checks the presence of juridical terms.

– **JUR:** $\forall j \in J, \forall t \in T(S)$, if $t = j$, mark t as juridical jargon.

The set J of juridical terms used in Learn PAd is composed of 877 terms in total. To compose this set, we have merged comprehensive glossaries selected from the Web. In particular, we have merged juridical terms from (a) the glossary provided by NY-COURTS.GOV, the New York State Unified Court System[4], (b) the glossary provided by the Judicial Branch of the State of Connecticut [5], and (c) the list of legal Latin terms in Wikipedia [6].

[4] `http://www.nycourts.gov/lawlibraries/glossary.shtml`
[5] `http://www.jud.ct.gov/legalterms.htm`
[6] `https://en.wikipedia.org/wiki/List_of_legal_Latin_terms`

4.3 Indicator: Difficult Jargon

This indicator quantifies the amount of sentences using terms (single and multi-words) that are considered difficult, either because they are rare, or because they are overly complex expressions that can be substituted with simpler ones. The Dale-Chall formula [3] measures the readability of a text by taking into account the percentage of words in the text not included in a list of $3,000$ words considered easy-to-read. Such formula has two primary defects in our context: (1) It gives only an index and does not indicate the editor which term is defective, i.e., hard to read; (2) the set of $3,000$ words is too restricted and risks to raise too many warnings. Indeed, a 5-6 years old child normally already uses $2,500$-$5,000$ common words [26], and by age 9, people normally build the set of words that they use every day. This set is normally composed of two sub-sets, a primary set (around $5,000$ terms), and a secondary set (around $10,000$ terms). Though also the secondary set includes terms that are used in every day life, such set includes also terms that are less common, and, hence, more difficult. Therefore, to identify the usage of difficult jargon, we define a pattern that, for each sentence, checks that each term is contained in the primary set. More formally, let S be a sentence, and let $W(S)$ be any word in the sentence. Moreover, let E be the set of $5,000$ terms that belong to the primary set of easy-terms. The following pattern checks the presence of difficult jargon:

- **DIF-1:** $\forall w \in W(S)$, if $w \notin E$, mark w as difficult jargon.

If a sentence has at least one word that is detected to be difficult, according to the previous pattern, such sentence will be marked as defective. As set E, we have used the set of top-5000 most common terms available at [7].

The previous pattern checks that terms used in a sentence are easy-to-read for a general public, and it is domain independent. Indeed, the list of common words is based on the selection of the most frequent words in genre-balanced corpus [7]. To detect difficult expressions that are *specific* of PA documents, we resort to use the list of pompous terms that litter official writing [21]. Such list of terms has been edited by the Plain English Campaign[7], with the objective of making official writing easier to read. While the list of easy words include only single-word terms, this list includes also multi-word terms (e.g., "acquaint yourself with", "despite the fact that", *etc.*). Therefore, we define a pattern to check the presence of difficult jargon according to such list. Let D be the set of difficult terms. Let S be a sentence, and let $T(S)$ be any sequence of words in the sentence. The pattern is as follows:

- **DIF-2:** $\forall d \in D, \forall t \in T(S)$, if $t = d$, mark t as difficult jargon.

If a sentence has at least one term that is detected to be difficult according to one of the previous patterns, such sentence is marked as defective. As set D, we have used the mentioned set of 407 difficult terms listed in [21].

[7] `http://www.plainenglish.co.uk`

5 Quality Attribute: Clarity

The *content clarity* quality attribute defines the degree of clarity of a BP description. Clarity of content is associated to specific aspects of sentences that make them more *understandable* from the procedural point of view. In other terms, this attribute focuses on aspects associated to the applicability of a procedure, such as the presence of well-defined actors in a sentence, and the presence of clear time constraints. The following sections describe the indicators that we consider for this attribute.

5.1 Indicator: Actor unclear

This indicator indicates that the actor of an action is unclear. This might occur in different cases, as e.g., in the following examples:

- *The **officer** shall send the review form within 5 days from the reception of the review request.*
- *The procedure shall be carried out before the end of March 2015.*

In the first case, it is unclear which officer is in charge of sending the review form. This situation might be resolved though the other sentences of the documents—where the concept of officer might be defined—, and can be apportioned to the cases of potential *pragmatic ambiguities* [13], not considered here. The second case, instead, is using the passive voice, and this is a typical case where the subject of the action, i.e., the actor, is not specified in the sentence, and he/she is therefore unclear. However, a simple "by" could help specifying the actor, as in the following rephrasing:

- *The procedure shall be carried out by the certification authority before the end of March 2015.*

In this section, we will define patterns to identify cases similar to the one shown in the second example. The pattern below has been defined such cases:

- **ACT:** (Auxiliary) (RegularPP | IrregularPP)+ (¬ "by")

The pattern matches any case where we have a term that indicates the presence of at least an auxiliary verb (i.e., "am", "are", "were", "being", "is", "been", "was", "be") followed by one or more past participle in regular form (i.e., any term terminating with "-ed") or irregular form (e.g., "written", "spent", "proven", etc. – a list of 175 irregular verbs have been used). Moreover, the pattern checks the presence of the preposition "by" following the verbs, as indicator of the potential specification of an actor.

5.2 Indicator: Unclear acronym

An acronym is word made from the initial letters or parts of other words, generally used to identify organisations (e.g., NATO, NASA, etc.) or domain specific concepts (e.g., BPMN, SQL, etc.). An acronym is normally composed of capital letters, which can be separated by full stops (e.g., F.A.O.), or not (e.g., FAO). This indicator checks for acronyms that are never expressed in their extended form (e.g., North Atlantic Treaty Organization for NATO). We have seen that undefined acronyms are a relevant problem in the real-world BP descriptions collected within the Learn PAd project [15]. Indeed, such BP descriptions include a large amount of sentences with acronyms, and in most of the cases the meaning of such acronyms is not defined in any part of the text. Though some acronyms are commonly used, many acronyms found are domain specific, or even procedure specific and need to be defined to clarify their meaning.

We define an algorithm that makes use of regular expressions to check the presence of unclear acronyms in a document. The algorithm first searches for potential acronyms (**Step 1**). Then scans the document to search for sentences where the potential acronym occurs together with its definition; if no sentence is found, the acronym is marked as unclear (**Step 2**).

Step 1 The following regular expression is used to find potential acronyms:

– **Find Acronyms:** $[A-Z|\backslash.]\{2,\}$

The expression matches any string of text with capital letters or full stops, if it is composed of at least two characters. This expression includes cases of sequences of full stops, and terms written in capital letters (e.g., "PROTOCOL" in a capitalized title). After the execution of the regular expression, these cases are discarded from the list of potential acronyms. In practice, all potential acronyms made of full stops are discarded, as well as sequence of capital letters longer than 5 character.

Step 2 In each sentence where the acronym appears, the algorithm checks if a sequence of words exist that express the acronym in its extended version. The following regular expression is used to find the presence of a potential extended version of an acronym of length "len" in a sentence. The value of "len" is computed without counting the full stops (CNR and C.N.R. have both len = 3).

– **Find Acronym Definition:** $([A-Z]+\backslash w+([\ \ ]|))\{\text{len}\}$

The regular expression searches for sequences of length "len". The sequences are required to be composed of one or more capital letters, followed by any word character ($\backslash w$), followed by a space ([ ]), or not (to detect final words). Finally, the algorithm checks that each capital letter in the matched string matches the capital letters found in the candidate acronym.

If the extended version of an acronym is found in at least one sentence in the document, the acronym is marked as `‘‘clear’’`, and no defect will be raised if

the acronym appears in the rest of the document without its extended version. If no sentence exist where the acronym appears together with its extended version, such acronym is marked as ‘‘unclear’’ in each sentence where the acronym appears. In turn, each sentence including an ‘‘unclear’’ acronym will be marked as defective.

6 Quality Attribute: Correctness

The *correctness* quality attribute defines the degree of grammatical correctness of a BP description. Hence, in this case, the quality attribute is equivalent to the indicator. Grammatical correctness is a fluid concept that evolves according to the evolution of a language and its grammar. Therefore, in our context, we have decided to give a more operational definition of correctness (i.e., a text is correct, if a grammar checker does not find any defect). To this end, we use a set of prescriptive rules, which are embedded in a tool, namely Language Tool[8], which has the advantage of embedding grammar checks that can be extended with the contributions of the user community. Therefore, as the grammar of a language evolves, we expect to easily plug additional patterns – or remove old ones –, so that the computed degree of correctness of a sentence is up-to-date with the rules of language.

7 The QuOD Tool

The different patterns have been implemented in the form of JAPE rules, deployed within a web service, and embeeded in the content analysis component of the Learn PAd platform [10, 15]. Furthermore, the QuOD web application has been implemented that, through RESTful APIs, interacts with the web service and allows users to check the quality of their BP descriptions and official documents in general. The web application was developed by Narwhal Software[9] and it is publicly available at `http://narwhal.it/quod/`.

Fig. 1 reports a screenshot of QuOD when applied to a sample BP description named EPBR (European Project Budget Reporting), see Thönssen et al. [24] for more details. On the top-left panel, the user can select the quality attributes to check (named Criteria), the Language, and the Document type. After performing the analysis, the system outputs a summary of the numerical scores associated to each quality attribute, indicating the percentage of defective sentences over the whole document for each attribute (bottom-left). On the right panel, the user can see the actual occurrences of the defects, highlighted with the color of the associated attribute. By hovering the mouse on the highlighted defect, the user can see the recommendation. For example, in the figure, we have an unclear actor in the sentence *[...] the authorization of the involved school has*

[8] `https://www.languagetool.org`
[9] `http://narwhal.it`

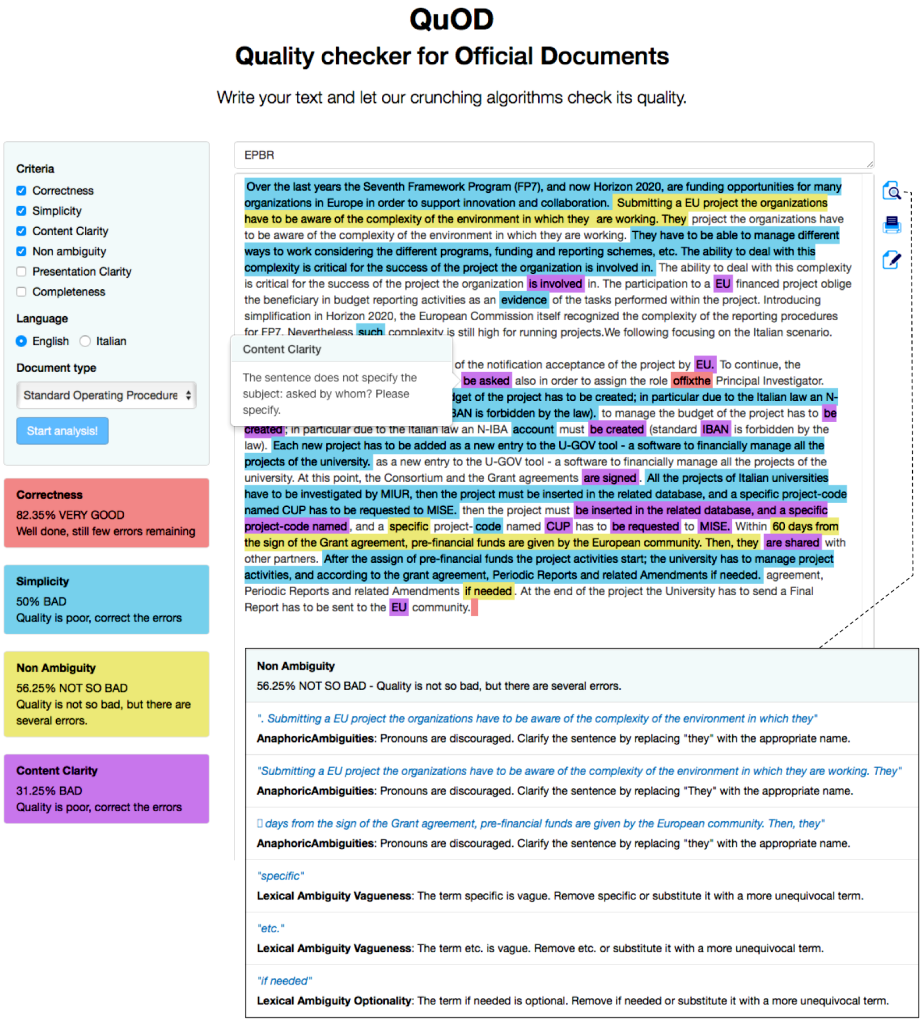

Fig. 1. The interface of QuOD when applied to a BP description.

to be asked [...], and we see a pop-up window with a recommendation concerning Content Clarity: *The sentence does not specify the subject: asked by whom? Please specify.*

By selecting the lens icon on the top-right corner, the user can also inspect the single defects. In the figure, we see the list of defects associated to the non-ambiguity attribute. This is particularly useful when overlapping defects are present in the original document, which may not be clearly visible in the central panel. For example, in the figure, we have two potential, and overlapping, anaphoric ambiguities: one is referred to the usage of "they" in the sentence *"Submitting a EU project the organizations have to be aware of the complexity of the environment in which* they *are working."*, identified with **AAMB-1** of Sect. 3.2. The other is referred to the usage of "They" in the following sentence, which refers to the previous one, and which was identified based on **AAMB-2** of Sect. 3.2.

8 Conclusion

Public administrations (PAs) typically use natural language to describe their business processes (BPs). As natural language is inherently ambiguous, descriptions of BPs need to be carefully reviewed for their linguistic quality. To support the work of editors and reviewers of BP descriptions in PAs, this paper presents QuOD, a tool oriented to detect linguistic quality defects in official documents in general, and in PA documents in particular. The tool is developed in the context of the EU project Learn PAd, and is publicly available through a web application. In the future, we plan to gather data from the users of the tool, and improve the defect detection capabilities to reduce *false positives*, as pattern-based systems are known to suffer from this problem [14]. A validation campaign is also foreseen with PA users, to assess and further improve the tool.

Acknowledgments

This work was possible thanks to the seminal work of Stefania Gnesi and coauthors on the usage of rule-based NLP techniques for detecting ambiguity and other quality issues in requirements specifications [17].

References

1. Berry, D., Gacitua, R., Sawyer, P., Tjong, S.F.: The case for dumb requirements engineering tools. In: Requirements Engineering: Foundation for Software Quality, pp. 211–217. Springer (2012)
2. Berry, D.M., Kamsties, E., Krieger, M.M.: From contract drafting to software specification: Linguistic sources of ambiguity (2003)
3. Chall, J.S., Dale, E.: Readability revisited: The new Dale-Chall readability formula. Brookline Books (1995)

4. Chinosi, M., Trombetta, A.: Bpmn: An introduction to the standard. Computer Standards & Interfaces **34**(1), 124–134 (2012)
5. Corradini, F., Ferrari, A., Fornari, F., Gnesi, S., Polini, A., Re, B., Spagnolo, G.O.: A guidelines framework for understandable bpmn models. Data & Knowledge Engineering **113**, 129–154 (2018)
6. Cutts, M.: The plain English guide. Oxford University Press (1996)
7. Davies, M.: Word frequency data. http://www.wordfrequency.info/free.asp, accessed: 1 Aug. 2015
8. De Angelis, G., Ferrari, A., Gnesi, S., Polini, A.: Collaborative requirements elicitation in a european research project. In: Proceedings of the 31st Annual ACM Symposium on Applied Computing. pp. 1282–1289. ACM (2016)
9. De Angelis, G., Ferrari, A., Gnesi, S., Polini, A.: Requirements elicitation and refinement in collaborative research projects. Journal of Software: Evolution and Process **30**(12), e1990 (2018)
10. De Angelis, G., Pierantonio, A., Polini, A., Re, B., Thönssen, B., Woitsch, R.: Modeling for learning in public administrationsthe learn pad approach. In: Domain-Specific Conceptual Modeling, pp. 575–594. Springer (2016)
11. Dell'Orletta, F., Montemagni, S., Venturi, G.: Read-it: Assessing readability of italian texts with a view to text simplification. In: Proceedings of the second workshop on speech and language processing for assistive technologies. pp. 73–83. Association for Computational Linguistics (2011)
12. Ferrari, A., DellOrletta, F., Esuli, A., Gervasi, V., Gnesi, S.: Natural language requirements processing: a 4d vision. IEEE Software **34**(6), 28–35 (2017)
13. Ferrari, A., Gnesi, S.: Using collective intelligence to detect pragmatic ambiguities. In: Requirements Engineering Conference (RE), 2012 20th IEEE International. pp. 191–200. IEEE (2012)
14. 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 Software Engineering pp. 1–50 (2018)
15. Ferrari, A., Spagnolo, G.O., Witschel, H.F.: Learn PAd - Deliverable D4.2 Quality Assessment Strategies for Contents (Apr 2019). https://doi.org/10.5281/zenodo.2643293, `https://doi.org/10.5281/zenodo.2643293`
16. Ferrari, A., Witschel, H.F., Spagnolo, G.O., Gnesi, S.: Improving the quality of business process descriptions of public administrations: resources and research challenges. Business Process Management Journal **24**(1), 49–66 (2018)
17. Gnesi, S., Lami, G., Trentanni, G.: An automatic tool for the analysis of natural language requirements. IJCSSE **20**(1) (2005)
18. Hirst, G.: Semantic interpretation and the resolution of ambiguity. Cambridge University Press (1992)
19. Leopold, H., Smirnov, S., Mendling, J.: On the refactoring of activity labels in business process models. Information Systems **37**(5), 443–459 (2012)
20. Massey, A.K., Rutledge, R.L., Anton, A., Swire, P.P., et al.: Identifying and classifying ambiguity for regulatory requirements. In: Requirements Engineering Conference (RE), 2014 IEEE 22nd International. pp. 83–92. IEEE (2014)
21. Plain English Campaign: The A to Z of alternative words. http://www.plainenglish.co.uk/files/alternative.pdf
22. Sanne, U., Ferrari, A., Gnesi, S., Witschel, H.F.: Ensuring action: Identifying unclear actor specifications in textual business process descriptions. In: Proceedings of the 8th International Conference on Knowledge Management and Information Sharing (KMIS), 2016. Springer Books (2016)

23. Silva, T.S., Thom, L.H., Weber, A., de Oliveira, J.P.M., Fantinato, M.: Empirical analysis of sentence templates and ambiguity issues for business process descriptions. In: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems". pp. 279–297. Springer (2018)
24. Thönssen, B., Witschel, H.F., Rusinov, O.: Determining information relevance based on personalization techniques to meet specific user needs. In: Business Information Systems and Technology 4.0, pp. 31–45. Springer (2018)
25. Trudeau, C.R.: The public speaks: An empirical study of legal communication. The Scribes J. Leg. Writing **14**(2011-2012) (2012)
26. UK Government: Content design: planning, writing and managing content. `https://www.gov.uk/guidance/content-design/writing-for-gov-uk`, accessed: 1 Aug. 2015
27. University of Sheffield,: Jape: Regular expressions over annotations. https://gate.ac.uk/sale/tao/splitch8.html, accessed: 1 Aug. 2015
28. Yang, H., Roeck, A.N.D., Gervasi, V., Willis, A., Nuseibeh, B.: Analysing anaphoric ambiguity in natural language requirements. Requir. Eng. **16**(3), 163–189 (2011)