Pragmatic Ambiguity Detection in Natural Language Requirements

Alessio Ferrari, Giuseppe Lipari, Stefania Gnesi, Giorgio O. Spagnolo
ISTI-CNR, Pisa, ITALY
⟨name⟩.⟨surname⟩@isti.cnr.it

***Abstract*—This paper presents an approach for pragmatic ambiguity detection in natural language requirements. Pragmatic ambiguities depend on the context of a requirement, which includes the background knowledge of the reader: different backgrounds can lead to different interpretations. The presented approach employs a graph-based modelling of the background knowledge of different readers, and uses a shortest-path search algorithm to model the pragmatic interpretation of a requirement. The comparison of different pragmatic interpretations is used to decide if a requirement is ambiguous or not. The paper also provides a case study on real-world requirements, where we have assessed the effectiveness of the approach.**

I. Introduction

Technical requirements are the foundation of most of the development activities of a software process. Considering that requirements have to be interpreted by different stakeholders of the process, these are normally provided in natural language [1], since this is the most widely understood communication code. Nevertheless, natural language is inherently ambiguous, and so are the requirements edited in such a language. Ambiguous requirements might cause misinterpretations among stakeholders, and lead to several problems. For example, the developer or modeller might unconsciously decide a possible interpretation of the requirement, and translate such interpretation into the software or model. Therefore, he/she could design something that is different from what was intended in the requirements. Though these problems might be resolved in the validation phase, it is preferable to discover ambiguities as early as possible to avoid *unconscious disambiguation* phenomena [2], and costly re-works on the artifacts.

Works on ambiguity detection and resolution in natural language requirements have been performed in the past [2]–[14]. Part of the works are focused on the identification of typical ambiguous terms and constructions [2], [3], [6], [7], [11]. Other works address the ambiguities by translating the requirements into formal languages or models [9], [10], [14]. Finally, some works focus on the usage of natural language understanding methodologies [4], [5].

The usage of artificial intelligence and statistic techniques in requirements analysis has been questioned by several authors (see, e.g., [15], [16]). In particular, for ambiguity detection, Berry *et al.* [16] advocate the usage of clerical, rule-based tools to detect ambiguities that can be identified without producing false negatives (i.e., without leaving ambiguities undiscovered). If part of the ambiguity detection task can lead to false negatives, that part shall be left to the thinking analyst.

In our work, we wish to focus on the detection of *pragmatic ambiguities*. Such ambiguities derive from the *context* of the requirements, and, in particular, depend on the background knowledge of the readers. Since the possible backgrounds are uncountable, it is not possible to provide rule-based infallible tools to detect these ambiguities. However, we believe that the work of the thinking, but fallible, analyst can profit from the support of a tool that, though fallible, can provide a different point of view in the analysis.

The present work relies on the approach presented in [17]. The approach models the background knowledge of the readers of a requirements document by means of graphs, which are automatically built from documents belonging to the domain of the requirements. The approach models the interpretation of a requirement as a shortest-path search in the graphs. The paths resulting from the traversal of each graph are compared and, if their overall similarity score is lower than a given threshold, the requirements specification sentence is considered ambiguous from the pragmatic point of view. This paper recalls the principles of the previous paper (Sect. III-A), and extends the approach with a novel algorithm to model the interpretation of a requirement (Sect. III-B). Moreover, it provides an approach to evaluate the adequacy of the domain documents employed to build the graphs, and to practically perform the evaluation of the pragmatic ambiguity (Sect. IV). Finally, the paper illustrates a case study in which we have experimented the current approach, and reached a precision of 51% and a recall of 63% (Sect. V). The current implementation of the techniques described in this paper is publicly available at `https://github.com/isti-fmt-nlp/Pragmatic-Ambiguity-Detector`.

II. Overview

The approach presented in this paper aims at discovering pragmatic ambiguities in natural language requirements documents. Roughly, a pragmatic ambiguity occurs in a requirement if different readers give different interpretations to it, depending on the *context* of the requirement. The context of a requirement includes the other requirements of the same document, which influence the understanding of the requirement, and the *background knowledge* of the reader [17], which gives a meaning to the concepts expressed in the requirement.

To understand the types of ambiguities that we are interested in, let us consider some relevant examples taken from our case study. The case study concerns a publicly available requirement specification of a system for Outbreak Management (OM) issued by the *Public Health Information Network* (PHIN) [18]. Such a system performs data collections from people that might be affected by epidemic health events. Data such as names of people, vaccinations, and clinical samples are aggregated by the system in order to control the outbreak. The examples that we wish to discuss are the following:

- 2.1.4 - Systems supporting OM should be able to electronically record and store data from *remote devices* that may be uploaded to an aggregating system;
- 2.3.1.1.a - Electronic questionnaires should provide the capability to accept *digital signatures*;
- 2.3.1.2 - Systems supporting OM must provide the ability to *control the configuration* of and revisions to investigation-specific questionnaires;

The first requirement uses the term *remote device*. This term might indicate different devices – e.g., laptops, smart-phones, tablets – which might need different drivers and appropriate software to be supported. Without a clear definition of the supported devices, different developers might produce different systems. If one is confident with smart-phone, s/he will focus on these systems, and ignore the possibility that data might come also from laptop computers.

The second requirement includes the term *digital signature*. With this term, one might intend different concepts, such as: a signature that has been certified by an independent authority; a signature to be provided through a digital device; the usage of a unique key for an authentication algorithm. Also in this case, a developer who reads the requirement is likely to choose the interpretation that is closer to his background on authentication systems.

The third requirement asks to *control the configuration* of questionnaires. This expression is too vague: it is unclear what kind of controls shall be performed. A reader might expect a validation with respect to the structure of such questionnaires. Another reader might expect a validation of the data included in the questionnaires.

Overview of the Approach The approach to discover pragmatic ambiguities that we present in this paper is depicted in Fig. 1. As informally exemplified, the pragmatic interpretation of a requirement depends on the *background knowledge* of a reader. Therefore, to emulate this dependency, our approach starts by building artificial subjects that model the background of different readers in the domain of the specification. To this end, artificial subjects are created in the form of domain knowledge graphs (**Domain Knowledge Graph Construction**). Given a set of documents in the domain of the requirements, a knowledge graph is a weighted graph that includes the concepts of the domain documents, and the relationships among concepts. Since we wish to model different artificial subjects, we will use different documents sets to produce different knowledge graphs. In our approach, we retrieve the documents by means of Web search engines, which are queried with domain-specific words chosen by the analyst.

An interpretation of a requirement can be regarded as a traversal of a knowledge graph according to the concepts included in the requirement (**Requirement Interpretation**). The traversal will activate concepts in the graph that are related to those in the requirement. The concepts traversed in a knowledge graph form the *pragmatic interpretation* of a requirement.

We now compare the pragmatic interpretations of each knowledge graph (**Interpretation Comparison**). Since the graphs are different, also the activated concepts will be different. However, for those requirements that use concepts that are well-established in the domain, we expect the pragmatic interpretations to be similar. In case a requirement uses concepts that can have different interpretations in the domain, the concepts that form the pragmatic interpretations are likely to be different. Therefore, in these cases, a pragmatic ambiguity might occur.

To model the usage of different groups of artificial subjects, we resort to provide different combinations of domain documents and repeat the process of analysis multiple times. Then, we compare the different results to come to a final value of common understanding of a requirement, which allows us to mark it as "`ambiguous`" or "`not ambiguous`".

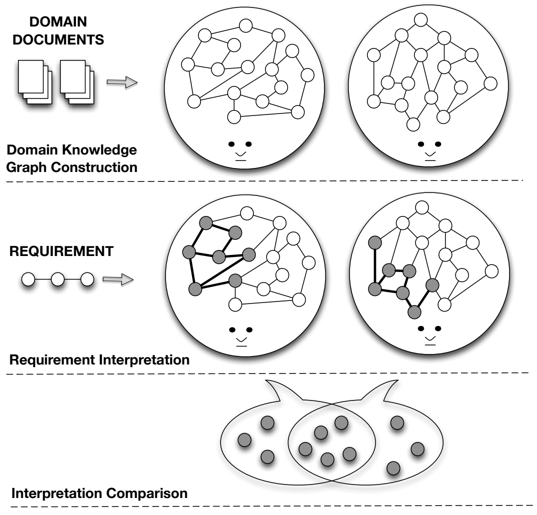
DOMAIN DOCUMENTS
Domain Knowledge Graph Construction
REQUIREMENT
Requirement Interpretation
Interpretation Comparison

Fig. 1: Overview of the approach

III. Core Algorithms

A. Domain Knowledge Graphs

The domain knowledge of an artificial subject is represented by means of a directed weighted graph $G = \{V, E, w\}$. Each node $v \in V$ is associated to a concept. Each edge $e \in E \subseteq \{V \times V\}$ expresses a relationship among two concepts. The weighting function $w : E \rightarrow [0, 1]$ is the inverse function of the semantic relatedness of two concepts. The more two concept are related, the lower the value returned by the weighting function. Such a graph is built from a domain documents set retrieved by means of Web search engines, as we will explain in Sect. IV-A.

1) Nodes: Each node is labeled with the morphological root of a relevant term contained in the document set. The relevant terms are those terms that are not part of the so-called *stop-words*, i.e., words which are common in English such as pronouns, prepositions, articles and conjunctions.

The use of morphological roots (also called *stems*) implies that terms such as "manage" and "management" are both represented by the same node "manag". In our case, the usage of morphological roots is based on the assumption that terms that share the same morphological root are normally related to the same high-level concept, and therefore a single node can be used to represent them.

More formally, given the set of documents D, let $T = \{t_1, t_2, \ldots, t_{|T|}\}$ be the set of unique terms in D, i.e., the *vocabulary* of terms, and let $\widehat{T} = T \setminus \Omega$, where Ω is the set of English stop-words.

Moreover, let M be the set morphological roots, and let $\phi : T \to M$ be the stemming function [19] that maps each term into its morphological root.

The set V of nodes in the graph is labeled with the set of distinct morphological roots of those terms that are not stop-words. Formally, given the set of stems which are not stop-words $\overline{T} = \{\phi(t) \mid t \in \widehat{T}\}$, we define a labeling function $\mathcal{L} : V \to \overline{T}$ that associates each node to a stem.

We argue that representing concepts as morphological roots of terms, instead of synsets as in Wordnet, is not too weak in our context. Indeed, the domains that are involved in the graph construction are technical domains, where a reduced vocabulary with a limited amount of synonyms is typically used.

2) Edges: Each directed edge in the graph is associated with the *co-occurrence* of two terms in the document set. We say that two terms co-occur when they appear next to each other in a sentence. Co-occurrence normally indicates semantic proximity among concepts [20], and therefore we used it to represent concepts relationships. The nodes of our graph are not labeled with terms, but with stems. Hence, the concept of co-occurrence is in this case related to stems.

More formally, let the sentences $S \in D$ be represented as an ordered list of its stemmed terms excluding stop-words, i.e., $\overline{S} = (\phi(s) \mid s \in S, s \notin \omega)$. Two stems $\bar{t}_i, \bar{t}_j \in \overline{T}$ co-occur in D *iff* $\exists S \in D, \exists \bar{s}_h, \bar{s}_k \in \overline{S} : \bar{t}_i = \bar{s}_h, \bar{t}_j = \bar{s}_k, k = h + 1$.

Therefore, edges $e = (v_i, v_j) \in E$ in our graph are all those couples of nodes such that $\mathcal{L}(v_i), \mathcal{L}(v_j)$ co-occur in the document set. The direction of each edge represents the order in which the stems connected by the edge can appear in a sentence of the document set. For example, if we consider a sentence such as S = *"The system shall provide a database to store the names of the patients"*, its stemmed version without stop-words is $\overline{S}$ = {'system', 'provid', 'databas', 'store', 'name', 'patient'}.

The co-occurrences contained in $\overline{S}$, and therefore the corresponding edges in the graph, are {('system', 'provid'), ('provid', 'databas'), ('databas', 'store'), ('store', 'name'), ('name', 'patient')}.

The direction of the edges is implicitely represented by the order of the terms, therefore an edge such as ('store', 'name') is distinguished from ('name', 'store').

3) Weighting Function: According to the definition of co-occurrence among stems given above, we define the co-occurrence frequency function $\chi : \overline{T} \times \overline{T} \to \mathbb{N}$, which associates to each couple of stems the number of their co-occurrences in the document set.

Each edge is labeled with a real positive number according to the weighting function w. Given an edge $e = (v_i, v_j) \in E$, the function is computed as $w(e) = 1/\chi(\mathcal{L}(v_i), \mathcal{L}(v_j))$. It is worth noting that the function is always defined, i.e., $\chi(\mathcal{L}(v_i), \mathcal{L}(v_j)) > 0$, since an edge exist in the graph if and only if the two stems that are connected by such edge co-occur in the original document set.

A domain knowledge graph G is automatically built from a set of documents D that belong to the domain of the requirements specification. The details of the algorithms are given in a previous paper [17]. If we choose different sets of domain documents, namely $D^1, D^2, \ldots, D^n$, we will have n different domain knowledge graphs. Indeed, the documents sets naturally contain different co-occurrences. However, we expect to have the same connections in the graphs for those concepts that are well defined in the given domain, since these are likely to occur together in all the documents sets. Moreover, we expect the connections to be stronger (i.e., to have lower weights) if the connection among these concepts is frequent in all the domain documents sets (i.e., the associated stems frequently occur together).

Consider for example the excerpt of two graphs taken from our case study, and centered on the term "patient" (Fig. 2). We see that terms highlighted in grey, namely "visit", "location" (stemmed as "locat"), "visit", "death" and "care" are connected to "patient" in both graphs: these connections are well established in the domain. The other connections, instead, are specific of each graph. Moreover, we see that the lowest value of the weight is achieved for the pair ("patient, care"), which appears to be a highly relevant connection in both graphs (the lowest values indicate the tightest connections).

B. Requirement Interpretation

When a person reads a sentence, s/he will activate concepts in her/his knowledge-base to perform interpretation of the sentence. The sentence is a sequential stream of words, and the concepts that are activated when reading it are those ones that are more related to the concepts sequentially expressed in the sentence. We argue that different pragmatic interpretations of a sentence by different subjects derive from the activation of different concepts in the knowledge-base of the subjects.

Back to our problem, we aim to detect pragmatic ambiguities in a requirements document. In our case, the knowledge-base of each subject is represented by a relational graph and the sentences that need an interpretation are those sentences contained in the requirements document. Activating related concepts in our graph implies traversing a minimum-weighted path from one concept in the sentence to the following concept

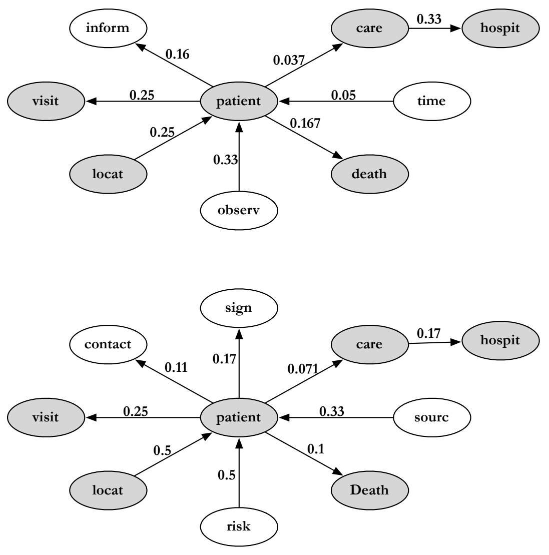

Fig. 2: Excerpts of domain knowledge graphs

in the sentence. Therefore, the interpretation of a requirement sentence can be regarded as the search of the *least-cost path* in the graph for each couple of neighbouring stems (i.e., concepts) in the input sentence.

To implement these arguments in the form of an algorithm, we resort to use an approach that employs a least-cost path search. Given a source node x, a goal node y and a weighted graph $G = \{V, E, W\}$, a least-cost search algorithm returns the least-weighted path between x and y. In our implementation, we have used the A^* [21] algorithm to compute the shortest path. The choice of A^* is driven by efficiency reasons. Comparison with other algorithms is left as future work.

The proposed approach works as follows. Given the sentences of our requirements document, these are pre-processed as previously performed on the domain-related documents. Each sentence in the requirement document is reduced to an ordered set of stems that are not stop-words, i.e., each requirement sentence R becomes $\overline{R}$, according to the previously introduced notation. Then, for each ordered couple $(\overline{r}_i, \overline{r}_j)$ of stems such that $\overline{r}_i, \overline{r}_j \in \overline{R}$ and $j = i + 1$, we perform a shortest-path search. We consider $\overline{r}_i$ as the source, and $\overline{r}_j$ as the goal. We build a path P as the conjunction of the shortest-paths found for each $(\overline{r}_i, \overline{r}_j)$. In the literature, a path like P is normally referred as a *connection subgraph* [22]. Finally, we extend the path P with all the nodes that are *directly* connected to the nodes in P. This extended path EP is returned by the algorithm. The choice of extending the minimum path has been performed to include in the interpretation process not only the concepts that appear in the shortest path, but also those concepts that can influence the interpretation of each single term of the path. In other words, we want to model the fact that, given the term "patient", a reader can think to "visit", "care", "death" and other terms that can be associated to "patient" and somehow contribute to its meaning.

The following pseudo-code illustrates the main procedure of our algorithm, named SENTENCE-PATH-SEARCH. The algorithm takes the pre-processed requirement $\overline{R}$, together with the knowledge graph elements, namely V, E, W and a vector X, which counts the number of co-occurrences for each edge (basically, the inverse of vector W).

```
SENTENCE-PATH-SEARCH($\overline{R}, V, E, W, X$)
    $P \leftarrow \emptyset$
    $V \leftarrow$ UPDATE-NODES($\overline{R}, V$)
    $P \leftarrow$ BUILD-MIN-PATH($\overline{R}, V, E, W, X$)
    $EP \leftarrow$ BUILD-EXT-PATH($P, V, E, W$)
    return EP
```

The UPDATE-NODES and BUILD-MIN-PATH have been formally reported in the previous paper [17]. Here, we informally describe the procedures, to better focus on BUILD-EXT-PATH(P, V, E, W), which is a novel contribution of the current work.

The UPDATE-NODES procedure simply adds nodes to V, if new stems are found in $\overline{R}$. This models the integration of additional nodes (i.e., additional knowledge) in the graph. In practice, we expect the number of added stems to be rather limited, since the knowledge base is expected to cover all the concepts included in the requirements.

The BUILD-MIN-PATH procedure performs the shortest path search for each couple of stems in $\overline{R}$, and implements additional graph modification steps. In particular, it creates additional edges if new pairs of stems are found in $\overline{R}$. Moreover, it decreases the weight of those edges that are traversed during the path search. This choice ensures that the traversal of concepts during requirements interpretation leaves a trace on the graph, by reinforcing the edges that connect concepts included in the sentence. The BUILD-MIN-PATH procedure returns a path P, which is the connection subgraph.

The BUILD-EXT-PATH(P, V, E, W) is implemented according to the following pseudo-code.

```
BUILD-EXT-PATH($P, V, E, W$)
    $EP \leftarrow \emptyset, ep \leftarrow \emptyset$
    for $p \in P$
        do
            $ep \leftarrow \emptyset$
            for $e \in E$
                do
                    if $p \in e$ and $W[e] < 1$
                      then $ep \leftarrow ep \cup e$
            $EP \leftarrow EP.ep$
    return $EP$
```

The $EP.ep$ notation stands for the conjunction between the currently built path EP and the newly retrieved sub-paths ep. Roughtly, the nodes in ep are attached to EP.

The procedure takes the connection subgraph P and the domain knowledge graph V, E, W as input. Then, for each node p in P, it retrieves all the nodes that are direct neighbours of p. These nodes, together with p, will form the extended subpath ep. The final extended path EP is built as the conjunction of all the extended sub-paths ep.

It is worth noting that the nodes in ep shall be connected to p by means of an edge e with a weight $W[e]$ that is strictly lower than 1. We have chosen to exclude nodes with weight equals to 1, since these have a quite weak connection with p. Indeed, the stems in these nodes co-occurred with p only once in the documents that have been used to build the knowledge graph. Therefore, we assume that these stems do not have a critical influence on the interpretation of the concepts associated to p.

C. Interpretation Comparison

Given a requirement document, each requirement sentence is processed by the SENTENCE-PATH-SEARCH procedure. We assume that during the domain knowledge graph construction phase, a set of n graphs has been generated by running the algorithm on n documents sets. Therefore, the least-cost path search is performed n times for each requirement sentence, each time employing a different graph.

Since the knowledge-bases are different, the generated paths are likely to be different. However, we can perform a comparison among paths to evaluate their similarity.

In order to compare the two paths, it is required to define a similarity metric. At this stage, we resorted to use an extension of the Jaccard similarity metric [23], defined as follows. Let $EP_1, EP_2, \ldots, EP_n$ be the different extended paths and let $\overline{EP}_1, \overline{EP}_2, \ldots, \overline{EP}_n$ be the label of the nodes (i.e., the stems) contained in the paths. Moreover, let $\widetilde{EP}_1, \widetilde{EP}_2, \ldots, \widetilde{EP}_n$ the stems in each path that have been added to the graphs by the UPDATE-NODES procedure.

The similarity of paths can be computed as:

$$\sigma_R(EP_1, EP_2, \ldots, EP_n) = \frac{|\bigcap_{i=1}^{n}(\overline{EP}_i \setminus \widetilde{EP}_i)|}{|\bigcup_{i=1}^{n}(\overline{EP}_i \setminus \widetilde{EP}_i)|}. \quad (1)$$

This similarity metric measures the proportion of stems that the paths have in common, and it assumes that the greater is the number of shared stems, the higher is the chance of the paths to be similar.

IV. AMBIGUITY EVALUATION

A. Domain Documents Gathering and Partitioning

A knowledge graph G is built from a set of domain documents D. Given the domain of the requirements specification, the set D of the the domain documents is searched through Web search engines. The search is performed using representative terms of the domain, which are chosen by the requirements analyst. The requirements analyst is also in charge of selecting the documents that are most relevant from those retrieved by the search engines.

The selection of the domain documents is a key step of the approach. Indeed, such documents will be employed to build the knowledge graph G, which represents the domain knowledge of the artificial subject that interprets the requirements. If the domain knowledge is weak, the artificial subject will not be able to give an interpretation to the requirement. Therefore, it is useful to provide a systematic approach for the requirements analyst to check the *coverage* of the domain documents with respect to the requirements document. Roughly, we define the *minimum coverage* as the amount of different stems included in the requirements document that are also included in the domain documents. More formally, given a requirements document $\mathcal{R}$ and a set of domain documents D, let $\bar{\mathcal{R}}$ and $\bar{D}$ be the sets of unique stems included in $\mathcal{R}$ and D, we define the minimum coverage index ρ of D with respect to $\mathcal{R}$ as:

$$\rho = \frac{|\bar{\mathcal{R}} \bigcap \bar{D}|}{|\bar{\mathcal{R}}|} \quad (2)$$

The minimum coverage index is employed by the analyst to evaluate if the document set is sufficient to build a knowledge graph that is representative of the knowledge of a reader. Normally, a value ρ that is as close as possible to 1 shall be achieved, which implies that all the stems included in the requirements are also included in the domain documents. If such value is not reached, the analyst identifies the words of the requirements that are missing from the domain documents. In practice, we have verified that missing terms can be partitioned into three categories:

common terms: if a term does not appear in the domain documents, it might imply that such term is not typical in the considered domain. Therefore, it is likely to be interpreted according to the common-sense knowledge of the readers, which might be different. Requirements including these terms have to be manually checked for potential ambiguities, and, in case, rewritten with more domain-specific words. For example, in our case study, we have the following requirement: *2.2.1.7 Systems supporting OM must have the capability to capture data about any public or private gathering of people (e.g., church social, ball game) [...].* The words "church","social", "ball", "game" are not typical of the outbreak management domain. In our case, we have decided to ignore the presence of these words since they were simply providing examples.

project-specific terms: the document might contain terms or acronyms that are typical of the project, and that might be defined in the input documents of the specification (e.g., preliminary requirements, taxonomy). In these cases, it is useful to include such input documents in D, or, if we do not have access to the input documents, we can discard these terms in the computation of the coverage. For example, in our case study, the acronym "OM" (Outbreak Management) was not found in any of the domain documents. Therefore, we ignored the term in the computation of the coverage.

domain-specific terms: these are terms that are typical of the domain, but are not included in the domain documents. In our case study, the words "chain" and "custody", which compose the expression "chain of custody" were not initially found in the domain documents. Therefore, we searched for additional documents to cover the expression "chain of custody".

The minimum coverage index simply checks that all the concepts in $\mathcal{R}$ are covered by the concepts in D. Other coverage indexes can be foreseen that take into account also the number of occurrences of a concept in D, and the number of connections with other concepts in G. However, within our framework, we have seen that the minimum coverage index is sufficient to check the adequacy of the document set.

In our approach, we expect to have a set of n knowledge graphs $G^1, \ldots, G^n$. Therefore, n different domain documents sets, $D^1, \ldots, D^n$ are needed. Each D^i shall have a ρ that is as close as possible to 1, to provide a minimum coverage of the requirements. We recommend that, for each D^i, $0.98 \leq \rho \leq 1$. This range has been chosen after experimenting with different sets of domain documents. We have seen that, with a large amount of terms in the domain documents, if 2% of the terms of the requirements document are missing from the domain documents, we have no actual effect on the sentence interpretation step of the approach.

The content of each G^i with respect to the other knowledge graphs in $G^1, \ldots, G^n$ might highly affect the interpretation comparison step of the approach. Indeed, if the documents used to build the knowledge graphs are not "different enough", the graphs might provide similar interpretations, even though, in practice, the interpretations could be different. To reduce this effect, it is useful perform the analysis k times with different combinations of documents for each graph (or, if enough documents are available, with non-overlapping groups of different documents). In practice, given a set of domain documents $\mathcal{D}$, we first group the documents $d \in \mathcal{D}$ into n different subsets $D^1, \ldots, D^n$, such as each subset has a minimum coverage $\rho \geq 0.98$. Then, we perform other groupings of documents in $\mathcal{D}$ with the same constraint for the coverage, i.e., $\rho \geq 0.98$. Each requirement is therefore analysed k times, and each time a value of $\sigma_R(EP_1, EP_2, \ldots, EP_n)$ is provided. These values are taken into account to identify an aggregate value of σ_R, which is used to finally check the ambiguity of the requirement.

B. Ambiguity Evaluation

For each requirement $R_i \in \mathcal{R}$, each j-th analysis, $j = 1, \ldots, k$ gives a similarity value $\sigma^j_{R_i}(EP_1, EP_2, \ldots, EP_n)$. To simplify the notation, we will write $\sigma^j(R_i)$ to indicate this value. We combine these values to obtain an aggregate similarity value $\Sigma(R_i)$ that takes into account the different analyses. The value $\Sigma(R_i)$ will be compared with a threshold value T to check whether the requirement is ambiguous or not. If $\Sigma(R_i) \geq T$ the requirement is marked as "`ambiguous`". If $\Sigma(R_i) < T$ the requirement is marked as "`not ambiguous`". Two methods are employed to compute the value of $\Sigma(R_i)$:

- *average value*:

$$\bar{\sigma}(R_i) = \frac{\{\sigma^1(R_i) + \sigma^2(R_i) + \cdots + \sigma^k(R_i)\}}{k} = \Sigma(R_i) \quad (3)$$

- *minumum value*:

$$\sigma_{min}(R_i) = min\{\sigma_1(R_i), \sigma_2(R_i), \ldots, \sigma^k(R_i)\} = \Sigma(R_i) \quad (4)$$

The threshold value T is computed as the arithmetic mean of all the $\Sigma(R_i)$, for $i = 1 \ldots m$, where m is the number of requirements in $\mathcal{R}$:

$$T = \frac{\{\Sigma(R_1) + \Sigma(R_2) + \cdots + \Sigma(R_m)\}}{m} \quad (5)$$

By substituting $\Sigma(R_i)$ with the values $\bar{\sigma}(R_i)$ and $\sigma_{min}(R_i)$ the threshold values $\tau_{\bar{\sigma}}$ and $\tau_{\sigma_{min}}$ are computed. Values of $\bar{\sigma}(R_i)$ below the threshold $\tau_{\bar{\sigma}}$ detect the presence of an ambiguity in R_i according to the average value method. In the same way, values of $\bar{\sigma}(R_i)$ that are below the threshold $\tau_{\sigma_{min}}$ signal the presence of an ambiguity in R_i according to the minimum value method.

It is worth noting that also different methods can be used to compute the value of $\Sigma(R_i)$ and T (e.g., geometric mean, harmonic mean, median, mode). Here, we have listed the methods that we have used also in our experiment.

V. Experimental Evaluation

We evaluate the effectiveness of our proposed approach on a publicly available software requirements document named "Outbreak Management Functional Requirements", issued in 2005 by the *Public Health Information Network* (PHIN) of the *Centers for Disease Control and Prevention* (CDC) (available at `http://www.cdc.gov/phin/library/archive_2005/OM_RSv1.0.pdf`) [18]. The document includes 114 requirements (i.e., $m = 114$) and concerns a system for the management of outbreaks and other health-related events. The system supports data collection, monitoring and analysis of data of people that might be affected an outbreak, e.g., contacts, clinical and environmental samples, laboratory results, vaccinations. The goal of the system is to provide a support to contain the epidemic health event. We have chosen this document for our case study since there is a large amount of domain documents available in the internet that concern the topic of the specification, with a large variety of information. The amount and variety of the information is useful to provide proper knowledge graphs. Indeed, these shall be sufficiently comprehensive – i.e., they shall cover all the terms of the requirements with appropriate contexts –, and this requires a large amount of information. Moreover, the knowledge graphs are expected to be also different enough to enable a significant evaluation of the requirements, and this requires the information sources to be various.

From the requirements, we have manually tagged those that were containing pragmatic ambiguities. In particular, 43 requirements have been marked as ambiguous. Representative examples are those reported in Sect. II.

After the choice of the requirements document, we have searched and collected the domain documents. These documents have been retrieved through the Google and Google Scholar search engines. The key-words employed for the search and submitted to the search engines in different combinations are "outbreak", "management", "system", "PHIN", "guidelines", "architecture", "health", "case definition", "chain

TABLE I: Domain documents employed in the case study

ID	**Title**	**Link**
d_1	PHEMCE strategy	http://goo.gl/hYaipm
d_2	Application to clinical and Public Health Practice	http://goo.gl/hVVy1Y
d_3	Biodefense countermeasure Department of Defense	http://goo.gl/I6U0Ns
d_4	Wikipedia page for "Case Definition"	http://goo.gl/yPndtx
d_5	Wikipedia page for "Chain of Custody"	http://goo.gl/4uvTuc
d_6	Definition of "Chain of custody"	http://goo.gl/OUgcQd
d_7	Communicable disease outbreak plan	http://goo.gl/rV72wX
d_8	Foodborn outbreak management	http://goo.gl/pTlgp9
d_9	Guidelines for the investigation and control of outbreaks	http://goo.gl/Sv4Ebu
d_{10}	Practice guidelines of the infectious diseases	http://goo.gl/GjLvg2
d_{11}	Implementation guide ambulatory healthcare	http://goo.gl/qEiLGR
d_{12}	Management of scabies outbreaks	http://goo.gl/GUAbKS
d_{13}	Modeling information systems architectures di P. Grefen	http://goo.gl/j2E4Lx
d_{14}	Outbreak control	http://goo.gl/f0HC1h
d_{15}	Outbreak management guidelines for healthcare	http://goo.gl/EcYVEi
d_{16}	Surveillance and response in humanitarian emergencies	http://goo.gl/ybje6i
d_{17}	PHIN guide for syndromic surveillance	http://goo.gl/lEz8zw
d_{18}	PHIN messagging guide for syndromic surveillance	http://goo.gl/3AAXNE
d_{19}	Developing a management system: an overview	http://goo.gl/0l5sth
d_{20}	Industrial system 800xA system architecture	http://goo.gl/RSaBnD
d_{21}	System architecture and complexity	http://goo.gl/v44tC0
d_{22}	WHO guidelines for epidemic prearedness and response	http://goo.gl/PK9yn7
d_{23}	Wikipedia page for "Management System"	http://goo.gl/mgWfhh
d_{24}	Wikipedia page for "Outbreak"	http://goo.gl/LUQEWm
d_{25}	Wikipedia page for "Scabies"	http://goo.gl/fjYYrQ

TABLE II: Combinations of domain documents employed for each *k*-th analysis, together with the number of nodes $|V|$, edges $|E|$ and the ρ value of each graph.

k	G^1 / G^2	$\lvert V_{G^1}\rvert$ / $\lvert V_{G^2}\rvert$	$\lvert E_{G^1}\rvert$ / $\lvert E_{G^2}\rvert$	ρ_{G^1} / ρ_{G^2}
1	$d_1 d_3 d_5 d_7 d_9 d_{11} d_{13} d_{16} d_{17} d_{19} d_{20} d_{23} d_{25}$	7131	62265	0.99
	$d_2 d_4 d_6 d_8 d_{10} d_{12} d_{14} d_{15} d_{18} d_{21} d_{22} d_{24}$	5970	33325	0.98
2	$d_2 d_3 d_6 d_7 d_{10} d_{11} d_{15} d_{16} d_{17} d_{19} d_{22} d_{23}$	7383	49989	0.98
	$d_1 d_4 d_5 d_8 d_9 d_{12} d_{13} d_{14} d_{18} d_{20} d_{25} d_{20} d_{24}$	5826	46179	0.99
3	$d_6 d_7 d_{15} d_{22} d_{16} d_{23} d_1 d_9 d_{18} d_{25} d_8 d_{14} d_{24}$	6375	58736	1
	$d_2 d_{10} d_{17} d_3 d_{11} d_{19} d_5 d_{13} d_{20} d_4 d_{12} d_{20}$	6642	34882	0.98
4	$d_6 d_{22} d_{16} d_1 d_{18} d_8 d_{24} d_{10} d_3 d_{19} d_{13} d_4 d_{20}$	6914	46384	0.99
	$d_{15} d_7 d_{23} d_9 d_{25} d_{14} d_2 d_{17} d_{11} d_5 d_{20} d_{12}$	6400	49848	0.98
5	$d_{22} d_1 d_8 d_{10} d_{19} d_4 d_{15} d_{23} d_{25} d_2 d_{11} d_5$	6693	41735	0.99
	$d_6 d_{16} d_{18} d_{24} d_3 d_{13} d_{20} d_7 d_9 d_{14} d_{17} d_{20} d_{12}$	6550	53973	1

of custody". From the retrieved documents, we have selected the most relevant ones after a quick review of the content to assess that the documents were not out of scope. The total set of domain documents is composed of 25 items, which are listed in Table I.

Different combinations of domain documents have been provided to define the knowledge graphs. In particular, 10 combinations have been considered that allow to achieve a coverage above 98%. These combinations have been used to build the graphs for the different analysis. In our case study, each analysis uses 2 knowledge graphs (i.e., $n = 2$).

Five different analyses have been performed (i.e., $k = 5$), which were considered sufficient to identify proper thresholds. The combinations employed in the analyses, together with the values of coverage, are reported in Table II.

According to the five analyses, the thresholds $\tau_{\bar{\sigma}} = 0.3247$ and $\tau_{\sigma_{min}} = 0.2781$ have been computed. With these thresholds, we have identified 55 and 53 ambiguous requirements, respectively. The results have been compared with the manually tagged requirements. The approach correctly identified 25 out of 43 (with $\tau_{\bar{\sigma}}$) and 27 out of 43 (with $\tau_{\sigma_{min}}$) ambiguous requirements. To evaluate the results, we have considered the typical indexes of precision p and recall r [23].

The values of precision and recall for the different thresholds are reported in Table III. We see that a higher value for both indexes is achieved with the $\tau_{\sigma_{min}}$ threshold. However, the results are rather comparable, and, since our approach has been applied solely on this case study, we cannot conclude that one threshold shall be preferred to the other. During the application of the approach, it is probably useful to use both thresholds, and review the results obtained in both cases.

TABLE III: Precision and Recall for the different thresholds.

Threshold	p	r
$\tau_{\bar{\sigma}} = 0.3247$	45%	58%
$\tau_{\sigma_{min}} = 0.2781$	51%	63%

Finally, in order to compare the performance of our technique to a *baseline* approach, we resort to use a *random predictor*. This solution randomly marks requirements as "`ambiguous`" with a probability that matches the ratio λ of ambiguous requirements that have been manually identified by human assessors. According to this solution, both precision (p) and recall (r) evaluate to λ, which in our sample data set is equal to $43/114 \approx 38\%$. Therefore, our approach outperformed the random predictor by $\Delta p = 13\%$ and $\Delta r = 25\%$.

As highlighted by Berry *et al.* [16], a requirements analysis tool shall be tuned to favour recall over precision. Indeed, false negative cases, which decrease the recall, are harder to discover than false positive cases, which decrease the precision. Therefore, it is useful to review the negative cases of our study, to check directions for improvement of the approach. Below, we give some representative examples that we have found (possible improvements are reported in Sec. VI).

An example of false negative is: *"2.2.4.2.f - Demographic information should be collected about the investigator [...]"*. The example includes the word "investigator", which has been considered vague. However, the requirement includes several non-vague terms, and it is rather long. Therefore, in the computation of the ambiguity, we have seen that the ambiguity residing in this single term does not emerge, since our similarity metric considers all the terms in the sentence, without giving emphasis to those that are more ambiguous than others.

Another case is the following: *"2.5.13 - Mapping interfaces and data dictionaries must be clearly defined [...] "*. "Mapping interfaces" and "data dictionaries" are vague expressions. However, they are composed of words that are rather common in the domain documents. For example, a concept such as "data" has a solid set of concepts attached to it occurring in the different graphs, and this makes it well-established in the domain knowledge of the artificial subjects. Since the ambiguity resides in the composed term "data dictionaries", and since composed expressions are not included in the nodes, these ambiguities are hard to find for our method.

A final example is the following: *"Both the jurisdiction investigating the event and the jurisdiction reporting the cases and associated investigations must be captured. [...]"*. The term jurisdiction is considered ambiguous in the context of the specification. Indeed, the specification induces a model in which the mentioned entities are normally classified as "subjects", "objects", "locations" and other classes. No classification is provided for the term "jurisdiction". Therefore, we have considered such term as ambiguous. However, since in the domain documents, "jurisdiction" appears to be well-established, we are not able to detect such an ambiguity. We conclude that pragmatic ambiguities at the level of the domain document, and not at the level of the background knowledge of the reader, are also hard to identify with the current method.

VI. Conclusion

This paper presented a method to detect pragmatic ambiguities in natural language requirements documents. The method has been evaluated on a real-world requirements set, and the results are promising. Therefore, we aim to further develop the method to increase its precision and recall. In particular, we have observed that ambiguities often reside in single words or expressions. Our current similarity metric does not emphasize this aspect. We are currently working on other similarity metrics that take into account the degree of ambiguity of single words of the requirements. Another aspect that we wish to improve concerns the terms that we are using in the graphs. Indeed, we currently consider only single stemmed words, whereas in requirements documents it is typical to find multi-word terms (such as, e.g., "data dictionary" or "chain of custody"). We plan to include approaches such as those in [24] or [25] to identify these multi-word terms and include them in our graphs. Finally, also efficiency is currently an issue. The whole analysis lasted eight hours (about 250 seconds for each requirement), and optimization is needed. The bottleneck of the approach is the current implementation of the BUILD-EXT-PATH procedure, which is bounded by $O(|V_{max}|^3)$, where $|V_{max}|$ is the cardinality of the graph with the highest number of nodes. Approaches to store the stems of the neighbours of highly used concepts in a "smart" way are under development.

Acknowledgment

This work was partially supported by the LearnPAd FP7-ICT-2013.8.2 European Project.

References

[1] L. Mich, M. Franch, and P. N. Inverardi, "Market research for requirements analysis using linguistic tools," *Requir. Eng.*, vol. 9, no. 1, pp. 40–56, 2004.

[2] D. M. Berry, E. Kamsties, and M. M. Krieger, "From contract drafting to software specification: Linguistic sources of ambiguity," 2003.

[3] D. M. Berry and E. Kamsties, "The syntactically dangerous all and plural in specifications," *IEEE Software*, vol. 22, no. 1, pp. 55–57, 2005.

[4] L. Mich and R. Garigliano, "Ambiguity measures in requirements engineering," in *Proc. of ICS'00, 16th IFIP WCC*, 2000, pp. 39–48.

[5] N. Kiyavitskaya, N. Zeni, L. Mich, and D. M. Berry, "Requirements for tools for ambiguity identification and measurement in natural language requirements specifications," in *Proc. of WER'07*, 2007, pp. 197–206.

[6] S. Gnesi, G. Lami, and G. Trentanni, "An automatic tool for the analysis of natural language requirements," *IJCSSE*, vol. 20, no. 1, 2005.

[7] W. M. Wilson, L. H. Rosenberg, and L. E. Hyatt, "Automated analysis of requirement specifications," in *Proc. of ICSE'97*, 1997, pp. 161–171.

[8] L. Mich, "NL-OOPS: from natural language to object oriented requirements using the natural language processing system LOLITA," *NLE*, vol. 2, no. 2, pp. 161–187, 1996.

[9] V. Ambriola and V. Gervasi, "On the systematic analysis of natural language requirements with Circe," *ASE*, vol. 13, 2006.

[10] L. Kof, "From requirements documents to system models: A tool for interactive semi-automatic translation," in *Proc. of RE'10*, 2010.

[11] B. Gleich, O. Creighton, and L. Kof, "Ambiguity detection: Towards a tool explaining ambiguity sources," in *Proc. of REFSQ'10*, ser. LNCS, vol. 6182. Springer, 2010, pp. 218–232.

[12] H. Yang, A. N. D. Roeck, V. Gervasi, A. Willis, and B. Nuseibeh, "Analysing anaphoric ambiguity in natural language requirements," *Requir. Eng.*, vol. 16, no. 3, pp. 163–189, 2011.

[13] F. Chantree, B. Nuseibeh, A. N. D. Roeck, and A. Willis, "Identifying nocuous ambiguities in natural language requirements," in *Proc. of RE'06*, 2006, pp. 56–65.

[14] A. Cimatti, M. Roveri, A. Susi, and S. Tonetta, "Formalizing requirements with object models and temporal constraints," *SoSyM*, vol. 10, no. 2, 2011.

[15] K. Ryan, "The role of natural language in requirements engineering," in *Proc. of ISRE'93*, 1993, pp. 240–242.

[16] D. M. Berry, R. Gacitua, P. Sawyer, and S. F. Tjong, "The case for dumb requirements engineering tools," in *REFSQ*, ser. LNCS, vol. 7195. Springer, 2012, pp. 211–217.

[17] A. Ferrari and S. Gnesi, "Using collective intelligence to detect pragmatic ambiguities," in *Proc. of RE'12*, 2012, pp. 191–200.

[18] P. H. I. Network, *Outbreak Management Functional Requirements*, 2005. [Online]. Available: http://goo.gl/BEyzah

[19] J. B. Lovins, "Development of a stemming algorithm," *MTCL*, vol. 11, pp. 22–31, 1968.

[20] P. D. Turney and P. Pantel, "From frequency to meaning: Vector space models of semantics," *J. Artif. Intell. Res.*, vol. 37, pp. 141–188, 2010.

[21] P. Hart, N. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Trans. SSC*, vol. 4, no. 2, pp. 100–107, 1968.

[22] C. Faloutsos, K. S. McCurley, and A. Tomkins, "Fast discovery of connection subgraphs," in *Proc. of KDD'04*, pp. 118–127.

[23] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Addison-Wesley, 2005.

[24] R. Gacitua, P. Sawyer, and V. Gervasi, "Relevance-based abstraction identification: technique and evaluation," *Requir. Eng.*, vol. 16, no. 3, pp. 251–265, 2011.

[25] A. Ferrari, F. dell'Orletta, G. O. Spagnolo, and S. Gnesi, "Measuring and improving the completeness of natural language requirements," in *REFSQ*, ser. LNCS, vol. 8396. Springer, 2014, pp. 23–38.