Information and Software Technology 74 (2016) 183–193

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Terminological inconsistency analysis of natural language requirements

Janardan Misra*

Accenture Technology Labs, Bangalore, India

ARTICLE INFO

Article history:
Received 19 March 2015
Revised 5 November 2015
Accepted 9 November 2015
Available online 30 November 2015

Keywords:
Requirements analysis
Requirements management
Terminological inconsistency analysis
Alias identification
Entity disambiguation
Latent semantic analysis

ABSTRACT

Context: Terminological inconsistencies owing to errors in usage of terms in requirements specifications could result into subtle yet critical problems in interpreting and applying these specifications into various phases of SDLC.

Objective: In this paper, we consider special class of terminological inconsistencies arising from term-aliasing, wherein multiple terms spread across a corpus of natural language text requirements may be referring to the same entity. Identification of such alias entity-terms is a difficult problem for manual analysis as well as for developing tool support.

Method: We consider the case of syntactic as well as semantic aliasing and propose a systematic approach for identifying these. Identification of syntactic aliasing involves automated generation of patterns for identifying syntactic variances of terms including abbreviations and introduced-aliases. Identification of semantic aliasing involves extracting multidimensional features (linguistic, statistical, and locational) from given requirement text to estimate semantic relatedness among terms. Based upon the estimated relatedness and standard language database based refinement, clusters of potential semantic aliases are generated. Results of these analyses with user refinement lead to generation of entity-term alias glossary and unification of term usage across requirements.

Results: A prototype tool was developed to assess the effectiveness of the proposed approach for an automated analysis of term-aliasing in the requirements given as plain English language text. Experimental results suggest that approach is effective in identifying syntactic as well as semantic aliases, however, when aiming for higher recall on larger corpus, user selection is necessary to eliminate false positives.

Conclusion: This proposed approach reduces the time-consuming and error-prone task of identifying multiple terms which might be referring to the same entity to a process of tool assisted identification of such term-aliases.

1. Introduction

Software requirement specifications (SRS) are often expressed using unrestricted natural language (NL) owing to ease of (manual) modeling and comprehension [1]. There are several tools and techniques for managing requirements [2], however, few support detailed quality analysis of requirements in unrestricted NL [3,4]. In practice, therefore semantic requirements analysis to identify ambiguities and inconsistencies is dependent upon manually time consuming process of detailed review of requirement specifications.

Usage of technical and application domain specific terms in SRS plays an important role in design, validation, and testing phases of SDLC and inconsistencies in the choice of term-usage could result into difficulties in interpreting and using these specifications [5]. For example, when multiple terms are referring to same entity/concept or when an incorrect term is used with reference to standard domain knowledge. Such inconsistencies might arise if requirements have been evolving over long periods and/or if multiple users have contributed in related requirements independently without synchronization.

In this work, we consider the problem of *entity disambiguation* referring to a special class of terminological inconsistencies arising when multiple terms spread across a corpus of natural language text requirements may be referring to the same entity. For illustration, consider following requirements:

Req1: "Reporting subsystem should allow admin to enter detailed steps onto process log database." ...

Req227: "IDS should read the log-data from prlog db updated by the logging module."

* Tel.: +919886752855.
E-mail address: janardan.misra@accenture.com, janardanm@acm.org, janardanmisra@acm.org, janardan.research@outlook.com

http://dx.doi.org/10.1016/j.infsof.2015.11.006
0950-5849/

In the second requirement (Req227), it may not be immediately apparent that the term "prlog db" is a short-form of "process log database" and whether the terms "reporting subsystem" and "logging module" are referring to the same component. In case, terms "reporting subsystem" and "logging module" are indeed referring to the same component, it could lead to inconsistent design and incorrect functioning of the IDS (intrusion detection system) if these terms are considered different owing to incomplete analysis. When relying solely on manual analysis of requirements, it is possible that such ambiguities in terminological usage may remain undetected until late in design cycle.

Multiple terms referring to the same entity are known as *term-aliases* (or simply *aliases*) and *entity disambiguation* refers to the process of identifying these term-aliases. From design perspective, term aliasing indicates incomplete or inconsistent specifications and may result in inconsistencies in design decisions. For example, more than one class might result from the requirements given before – Class ReportingSys {…} with responsibility to allow admin to enter process step related data onto 'process log database' and classes related to the process IDS {…} interacting with Class LoggerModule{…}.

In our earlier work [6], we considered two variants of the problem of entity disambiguation in requirements. In first case, requirements may contain syntactic variances of terms, e.g., short-forms and acronyms. These syntactic variances might be difficult to identify using existing document search tools including regular expression based search, which would demand a user to know 'correct' patterns for relating variants with their original forms and then express these patterns using regular expressions to uncover such relationship among terms. Using proposed approach, a tool can assist users by generating lists of various syntactic variances of terms which might be present in the SRS spread across multiple documents.

Another variant is the case of semantic aliasing referring to scenarios where syntactically different terms have been used to refer to the same entity. According to [7], approximately 40% of errors in word usage (in plain English text) are of this type. However, no amount of syntactic pattern based search is sufficient to uncover these terms because inferring the relations among multiple terms referring to same entity or concept would require deeper semantic analysis. There are no existing document processing tools which provide such a feature where different terms with close semantic relatedness can be identified. Again proposed approach can be used to uncover potential semantic term aliases in the requirement specifications.

Current paper extends the solution approach presented in [6] in many ways including:

- Extended list of explicit alias patterns (ref. [Step5]).
- New technique to identify terms which are equivalent with respect to semantically neutral phrases (ref. [Step6]).
- Refined approach for semantic compression of term co-occurrence matrix (ref. [Step 9.2]) and tag-equivalence based compression (ref. [Step 10.2.2]).
- Extended approach for elimination of false positive results using WordNet based analysis (ref. [Step10.6.3] and [10.6.4]). In particular, with current approach, precision of the process of elimination of false positives for multi-word entity terms is expected to increase at the cost of reduction in recall of alias identification.
- Refinement over the process of generating alias clusters (ref. [Step11]). In particular, in [6], parameter *conTh* (used while merging clusters of closely related terms) was user selected parameter. With refined approach presented in this paper, *conTh* is determined automatically.

Rest of the paper is organized as follows: Section 2 presents background and related work. Section 3 discusses proposed solution approach in detail. Section 3.2 considers approach for identifying syntactic aliases as explicitly introduced ones, fuzzy variants, short-forms, and semantically equivalent phrases. Section 3.3 details steps to represent requirement corpus into vector space model followed by its transformation into latent semantic space and measurement of various similarities among terms and their clustering in order to bring semantically closely related terms together. An experimental analysis using a prototype tool is presented in Section 4. Section 5 concludes the paper.

2. Background

Problems related to terminological inconsistencies in SRS including automated entity disambiguation have received less attention in the domain of requirements engineering though discussed implicitly in [3,5].

Closely related problems, however, exist in other domains e.g., web search, social security, data mining etc. These problems include – entity resolution [8–10], alias detection [11], reference disambiguation [9], and data matching [9]. Key solution approaches include attribute analysis [8,9], link/reference analysis [10], term clustering [12], or combination of these [13]. See [8] and [9] for further details and additional background references on these.

However, most of the existing work has been in reference to specific types of entities e.g., name/ email/reference aliasing for author/person/conference as entities. Also approaches with good success rates tend to use supervised learning (tagged training data) to learn the patterns associated with referenced entities [11,14].

In contrast to the existing approaches, continuing with [6], we suggest a generic approach, which does not have limited focus on any specific type of entities while disambiguating their references. Also, it is not dependent on user inputs for patterns to be used for identifying term-aliasing. These constraints, however, render the problem inherently harder and not many works deal with such cases [15].

3. Solution approach

Let us start with defining terminology used in specific contexts in the paper: *Term* will be used to refer to a word or multi-word phrase. For example, 'reporting subsystem' and 'logging module' are terms. List of constituent words of a term would be enclosed within '{…}'. A *Language term* will refer to common language specific terms as occurring in standard language database e.g., WordNet [16]) in contrast to the *domain terms* having application specific interpretation. Among the domain terms we specifically consider *entity-terms*, which refer to entities in the requirement specifications.

Inputs: Set of natural language requirement documents: Each document may consists of one or more requirements in plain English. A requirement may consist of multiple sentences. Even though requirement specifications may span multiple documents, we process all the requirements together as if present in a single document itself.

Outputs: User guided iterative generation of glossary of entity term-aliases, which in turn could be used for terminological unification by selecting a representative term among the aliases and replacing all others by this term across the requirements. Alternately user could also search for potential aliases for an entity-term.

Fig. 1 presents high level schematic view of the overall process flow starting from extracting entity terms from given plain-test SRS to user selection of aliases. These process-step are described in the following sections.

3.1. Steps…generating term Corpus

[Step0] (Text processing) This step involves tasks commonly known as ETL (extract, transform, and load) involving extraction of document text and pre-existing structural information (sections, bullets etc.), detecting sentence boundaries, and cleaning the text corpus by unifying different types of brackets, collapsing multiple spaces with a single space, removing white spaces between beginning quote