Semantic Approach for Traceability Link Recovery using Uniform Resource Identifier (STURI)

Mazen Alobaidi, Khalid Mahmood
Department of Computer Science and Engineering
Oakland University, Rochester, USA
Email: malobaid@oakland.edu, mahmood@oakland.edu

Abstract—The efficiency and effectiveness of traceability link recovery in requirements management is becoming increasingly important within Requirement Engineering due to the complexity of project developments, such as continuous change in requirements, geographically dispersed project teams, and the complexity of managing the elements of a project - time, money, scope and people. Therefore, the traceability links among the requirements artifacts, which fulfill business objectives, is so critical to reducing the risk and ensuring the success of products. To that end, this paper proposes a semantic based traceability link recovery (STURI) architecture that presents the meaning of texts in Uniform Resource Identifier derived from Linked Data. To the best of our knowledge, this is the first architectural approach that uses Uniform Resource Identifier for finding similarities among requirements and among the automation of the recovery traceability.

Keywords—Semantic Web, NLP, Linked Open Data, Document Similarity, Uniform Resource Identifier

I. INTRODUCTION

The success of the project relies on robust requirement management tools. Requirement management is the most effective phase of project development. As a requirements management expert, Peter Zielczynski defines the major steps in requirements management as follows: establishing a plan, eliciting requirements, developing the vision document, creating use cases, supplementary specification, and finally a system design. Indeed, the backbone of requirements management is “Requirements Traceability” by which it is possible to map individual artifacts of requirements with other artifacts in the system. In addition, requirements traceability identifies and outlines the lineage of each requirement, apart from its backward traceability (derivation), its forward traceability (allocation) and its association to other requirements. Traceability, according to the IEEE Standard Computer Dictionary [1], is defined as the degree to which a relationship can be established between two or more artifacts (document) of the development process. Traceability is employed to guarantee solution conformance to requirements and to assist in scope and change management, risk management, time management, cost management, and communication management. Furthermore, it is used to distinguish missing functionality or to identify if implemented functionality is not supported by a specific requirement.

In current approaches, the relationship among requirement engineering artifacts is represented by their Semantic Similarity and Semantic Relatedness. Semantic Similarity is usually defined by considering the lexical relations of synonymy, or equivalent words and hyponymy, or the type-of relation. Semantic Relatedness, on the other hand, extends the definition of similarity by examining all types of semantic relations that connect two concepts. Such relations include, in addition to the above-mentioned two similarity relations, the antonym, metonymy, or the relations between wholes and parts, functional relations of frequent association as well as other non-classical relations. Although, enormous progress has been made in traceability link recovery using the current approaches, our proposed approach is more unique from the perspective of sharing information, this has been achieved by making use of a unique and uniform mechanism of identification, the Uniform Resource Identifier (URI) [3]. Our approach can be summed up in the twofold principle: considers all terms/words in a document as a resource and each resource uniquely identifies web resources.

The recovery/generation of the links of the traceability matrix, which utilizes URI among the artifacts, will be highly accurate as all nouns are compared with respect to URI rather than a key-word, synonymy, hyponymy, wholes and parts relation. One of the motivations of this project is to develop an automated traceability matrix for project managers using a Semantic based Traceability Link Recovery (STURI) framework that will use Uniform Resource Identifier, the Linked Open Data (LOD) [7], and Natural Language Processing [10] to find association among various artifacts of projects.

The paper is structured as follows. The next section is the related works. Section 3 contains the preliminary. In section 4, the concept, architecture and design of STURI are presented. The evaluation and results are shown and discussed in section 5, and the paper is concluded in section 6.

II. RELATED WORKS

There are two distinct disciplines of research that are associated to our proposed approach, namely Information Retrieval technique (IR) [11]–[13], and Semantic technique [25]–[27].
A. Information Retrieval Technique

Recovery traceability links has been extensively studied in Information Retrieval (IR). IR is mainly defined as discovering processes of nominee traceability links on the basis of the similarity between software engineer artifacts that can be transforming in some unstructured test format [14]. Antoniol, Giuliano et al. [14] proposed a method based on Information Retrieval (IR) to establish and maintain traceability links between source code and free text document where documents are ranked against queries constructed from the identifiers of source code. Ali et al. [16] proposed an approach to establish and maintain traceability links between source code, software requirements, and software repository requirements as well as improve the precision and recall of information retrieval (IR) techniques by showing that mining software repositories (MSR) and combining the mined outcomes with IR techniques improves the precision and recall of requirement traceability links. Marcus et al. [17] proposed approaches that advocate for the use of latent semantic indexing (LSI) to recover traceability links between documentation and source code. All the above proposed approaches based on IR, however, lack accuracy (precision, recall) [16], perform poorly in short context [18], and disregard word order, syntactic relations, morphology, semantic relation, and word ambiguities [19].

B. Semantic Technique

Semantic Similarity and Semantic Relatedness have recently received the attention of researchers who are studying traceability link recovery. Semantic Similarity is usually defined by considering the lexical relations of synonymy, or equivalent words, and hyponymy, or the type-of relation [20]–[22], [26], [27]. Semantic Relatedness, on the other hand, extends the definition of similarity by examining all types of semantic relations that connects two concepts [23]–[25]. Zhang, Witte et al. [19] proposed an approach which creates traceability links between source code and documentation software that can be summarized as follows: building ontologies, modeling the domains of source code and software documents, creating a knowledge base by automatically population these ontologies through code analysis and text mining, and finally establishing traceability links between code and documents through ontology alignment. Falbo et al. [28] contributed to this semantic approach by proposing an extended semantic document management platform for the requirement domain by using semantic annotations in requirement documents. Furthermore, there is an exploration of the conceptualization established by the proposed software requirements, with reference to its ontology and the generation of the traceability matrix, both of which are based on a dependency relationship and related axioms (reasons). Mahmoud et al. [29] also offered another appoint, based on semantic relatedness, which brings human judgment to an earlier stage of the tracing process by integrating it into the underlying retrieval mechanism. This would use a measure based on Wikipedia, namely Explicit Semantic Analysis. Although all the mentioned approaches improve accuracy, they have some limitations: only one database is used, which recovers traceability between source codes and software artifacts only, which in turn ignores word ambiguities.

III. Preliminary

A. Linked Data

The Web enables us to link related documents. Similarly it enables us to link related data. The term Linked Data refers to a set of best practices for publishing and connecting structured data on the Web. Key technologies that support Linked Data are URIs (a generic means to identify entities or concepts in the world), HTTP (a simple yet universal mechanism for retrieving resources, or descriptions of resources), and RDF (a generic graph-based data model with which to structure and link data that describes things in the world) [8].

B. Linking Open Data

Linking Open Data (LOD) project is to extend the Web with a data commons by publishing various open datasets as RDF on the Web and by setting RDF links between data items from different data sources.

---

Fig. 1. Semantic based Tractability Link Recovery Architecture (STURI)

IV. Semantic Based Tractability Link Recovery Architecture (STURI)

Due to the complexity of the product development within diverse industries, it is imperative that the recovery of traceability links in requirements management be efficient and effective. We propose a solution that derives document similarities from the Uniform Resource Identifier contained in the
compared Documents. Figure 1 shows the system architecture of components. Our architecture is comprised of three components. Namely: “Natural Language Processing (NLP)”, “URI extraction”, and “Inference Engine”. The following sections present a detailed description of each of the above components. Since semantic similarity between Documents is used URI semantic similarity, we will first describe our method for measuring URI semantic similarity.

A. URI semantic similarity

Approaches for measuring semantic similarity have been developed in the previous decade. Different similarity methods have proven to be useful in some specific applications: namely, artificial intelligence, natural language processing, information retrieval, and data mining, we proposed a document similarity measure which provides the greatest correspondence to common sense. Given two documents, document a and document b, we need to find the semantic similarity s(a,b). We can do this by implementing and evaluating using LOD as the underlying reference ontology, we get all URIs for each word in documents ‘a’ and ‘b’ and we measure the similarity based on our weighting scheme and overlapping URIs.

B. NLP module

The NLP module uses Apache Lucene framework [4] for uploading the artifacts, tokenization, stop words, stemming, and lemmatization [10]. Tokenization is the process that splits the artifacts into tokens, stop words are words which are filtered out, where stemming and lemmatization are the process of converting or removing the inflexional, derivational form to a common world form.

C. Uniform Resource Identifier Extraction module

The important tasks of this module are extracting RDF statements from LOD, building RDF store, building SPARQL query, executing query, URI disambiguation, and creating URI result map collocation. The process of this module is as follow:

1) Extraction RDF triples and building RDF graph store: to extract triples from LOD we download Dataset and read all RDF triples of the dataset. Also, we created RDF graph store using Jena tools.
2) Building SPARQL query: to find all the URIs of a resource, we dynamically construct SPARQL query that match all subjects of RDF triple with input resource term.
3) URI Disambiguation: We use the Naive Bayes classification technique [31] to classify the document and the dereferenced URI resource. In turn, we conducted a string match between classified classes of that document and resource; if there is a match, then we consider the URI as a candidate.
4) Creating URI result Map: to improve the performance of Inference engine, we constructing map collection to hold all the URIs of document.

Algorithm 1 shows the pseudo code of our implementation

D. Inference Module

The Inference module functionality maintains a count for the Overlapping of URI between two artifacts and the calculation of a “similarity score”. In addition, it generates candidates for traceability links. Figure 2 shows the URI semantic algorithm data structure. This module provides the following:

1) Reading all URIs of each term in the text
2) Building two space dimension matrix that represents terms in URIs
3) Creating graph models represent all terms that shares same URI
4) Recording the length of the path between nodes in a graph. The path represents the distance between two terms
5) Calculating the average length path for each graph
6) Creating URImap collection based on the average length path greater than or equal to URIThreshold
7) Using the URImap to measure the relatedness between two texts by calculating the URI overlapping

![Fig. 2. URI Semantic Algorithm Structure](image-url)
After the URI sets of two documents are formulated, semantic similarity between documents can be calculated by

$$\text{Sim}(\text{URIMAP}_a, \text{URIMAP}_b) = \frac{\text{URIMAP}_a \cup \text{URIMAP}_b}{\text{URIMAP}_a \cap \text{URIMAP}_b}$$

(V. Evaluation)

In order to measure the efficiency and the accuracy of the proposed framework, we implemented an experimental framework using General Architecture and Engineering Text Lucene and Jena which are Apache development environments that provide a rich set of interactive tools for the creation, measurement and maintenance of software components for processing human language. Ultimately, in order to simulate a real world scenario, we have selected three different datasets.

A. DataSets

The datasets used are illustrated below and see table I.

1) MODIS: The NASA Moderate Resolution Spectrometer (MODIS) dataset [30] is a small dataset created from the full specification (high- and low-level requirements documents) for the MODIS space instrument software. This dataset contains 19 high-level requirements, 49 low-level requirements, and a validated true positive 41 links that we refer to as the "True traces".

2) CM-1: The dataset consists of a complete requirement and a complete design document for a NASA space instrument [30]. The dataset contains 235 high level and 220 low-level requirements. The trace for the dataset was manually verified. The "theoretical true trace" (answerset) built for this dataset consisted of 361 correct links. Each of the high and low-level files contain the text of one requirement element.

3) Standard Company: A dataset of requirements management tools from Borland CaliberRM. The dataset contains 27 high level and 27 low-level requirements. The "theoretical true trace" (answerset) built for this dataset consisted of 19 correct links.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>High Req</th>
<th>Low Req</th>
<th>Traces</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODIS Dataset</td>
<td>19</td>
<td>48</td>
<td>39</td>
</tr>
<tr>
<td>CM-1</td>
<td>235</td>
<td>220</td>
<td>361</td>
</tr>
<tr>
<td>Standard Company</td>
<td>27</td>
<td>27</td>
<td>19</td>
</tr>
</tbody>
</table>

B. Experimental Results

In the experiment, the accuracy of our approach framework is compared against Vector Space Model [11] and Wu Palmer algorithm [27]. The primary accuracy measures used for comparison are Precision and Recall. Precision is the ratio of the number of true positive links retrieved over the total number of links retrieved where Recall is the ratio of the number of true positive links retrieved over the total number of true positive links.

We have implemented VSM, Wu Palmer and STURI algorithms as part of a requirement tracing tools that we have built called “Dynamic Trace Workbench”. We used the datasets mentioned above. To evaluate STURI approach against VSM and Wu Palmer, we have run two experiments.

1) First Experiment: We ran each algorithm mentioned above using all datasets. The results of the running experiments were collected and analyzed against existing answer sets and the results information were used to calculate Precision and Recall for evaluation. Figure 3, 4, and 5 compare the Recall measurement achieved by STURI and those for two benchmark techniques. The analysis of the result shows that STURI provides better Recall measurement cross all datasets. On the other hand, Figure 6, 7, and 8 shows that STURI preforms better than Wu Palmer algorithm in Precision, however VSM gives us better precision.

![Fig. 3. Recall for Standard Company Dataset](image1)

![Fig. 4. Recall for Modis Dataset](image2)
2) Second Experiment: The second experiment began by using the VSM algorithm against one of the datasets from above. We ran the STURI against the same dataset, and then we aggregated the two results and calculated the Recall and Precision. Moreover, we repeated the same experiment against the other two sets, as well as with Wu Palmer and STURI. The analysis of the result shows, on average, a correspondence of 30 percent on Recall. Recall improved on combining VSM/WU and STURI over running VSM or Wu Palmer alone. In addition, the results show on average a Precision improvement of 3 percent when combining VSM/WU and STURI over running VSM or Wu Palmer alone. In this experiment we presented a subset of the results, see Figure 9 and 10.
VI. CONCLUSION

These inter-industrial projects are constantly updated and modified in light of new risks and developing products. Traceability links are a vital part of requirements management for these industries. Since Automated Traceability Links are an important element to ensure the success of Software engineering projects, our proposed framework helps Software engineer projects to meet business requirements, to improve the precision and recall of traceability links between requirements artifacts, and increases the efficiency of time management.

REFERENCES


