Improving Traceability Links Recovery in Process Models Through an Ontological Expansion of Requirements

Raúl Lapeña[1(✉)], Francisca Pérez[1], Carlos Cetina[1], and Óscar Pastor[2]

[1] SVIT Research Group, Universidad San Jorge, Zaragoza, Spain
{rlapena,fperez,ccetina}@usj.es
[2] Centro de Investigación en Métodos de Producción de Software, Universitat Politècnica de València, Valencia, Spain
opastor@pros.upv.es

Abstract. Often, when requirements are written, parts of the domain knowledge are assumed by the domain experts and not formalized in writing, but nevertheless used to build software artifacts. This issue, known as tacit knowledge, affects the performance of Traceability Links Recovery. Through this work we propose LORE, a novel approach that uses Natural Language Processing techniques along with an Ontological Requirements Expansion process to minimize the impact of tacit knowledge on TLR over process models. We evaluated our approach through a real-world industrial case study, comparing its outcomes against those of a baseline. Results show that our approach retrieves improved results for all the measured performance indicators. We studied why this is the case, and identified some issues that affect LORE, leaving room for improvement opportunities. We make an open-source implementation of LORE publicly available in order to facilitate its adoption in future studies.

Keywords: Traceability Links Recovery · Business Process Models · Requirements Engineering

1 Introduction

Traceability Links Recovery (TLR) has been a subject of investigation for many years within the software engineering community [11,21]. Traceability can be critical to the success of a project [25], leads to increased maintainability and reliability of software systems [10], and decreases the expected defect rate in developed software [16]. However, TLR techniques rely greatly on the language of the studied documents. Often, when requirements are written, parts of the domain knowledge are not embodied in them, or embodied in ambiguous ways. This phenomena is known as tacit knowledge. The tacit knowledge is assumed by all the domain experts, and never formalized in writing. This behavior has been

P. Giorgini and B. Weber (Eds.): CAiSE 2019, LNCS 11483, pp. 261–275, 2019.
https://doi.org/10.1007/978-3-030-21290-2_17

reported by previous works [3,22]. As a result, both the text of the requirements and the tacit knowledge are used to build software artifacts, which in turn contain elements that are related to the text of the requirement, and elements that are related to the tacit knowledge. However, since part of the knowledge is not reflected in the text of the requirement, recovering the most relevant software artifact for a requirement through TLR becomes a complex task.

Through this work, we propose LORE, a novel approach that minimizes the impact that tacit knowledge has on TLR. To that extent, Natural Language Processing (NLP) techniques are used to process the requirements, and then an ontology is used to expand the processed requirements with concepts from the domain. Finally, TLR techniques are applied to analyze the requirements and software artifacts in search for software artifact fragments that match the requirements. We have evaluated our approach by carrying out LORE between the requirements and process models that comprise a real-world industrial case study, involving the control software of the trains manufactured by our industrial partner. Results show that our approach guides TLR to enhanced results for all the measured performance indicators, providing a mean precision value of 79.2%, a mean recall value of 50.2%, a combined F-measure of 66.5%, and an MCC value of 0.62. In contrast, the baseline used for comparison presents worse results in these same measurements. Through our work, we have also identified a series of issues related to the ontology and the requirements that prevent our approach from achieving better solutions. These issues could be tackled in the future to further improve the TLR process between requirements and process models.

Through the following pages, Sect. 2 presents the background for our work. Sections 3 and 4 provide details on our approach, and on the leveraged Traceability Links Recovery technique. Section 5 describes the evaluation of our approach. Section 6 introduces the obtained results. Section 7 discusses the outcomes of our work. Section 8 presents the threats to the validity of our work. Section 9 reviews works related to this one. Finally, Sect. 10 concludes the paper.

2 Background

In industrial scenarios, companies tend to have a myriad of products with large and complex models behind, created and maintained over long periods of time by different software engineers, who often lack knowledge over the entirety of the product details. Through this section, we provide an overview of the models in our case study, and of the problem that our approach intends to mitigate.

2.1 Case Study Models

Figure 1 depicts one example of a model, taken from a real-world train, specified through a process model. The model has the expressiveness required to describe the interaction between the main pieces of equipment installed in a train unit, and the non-functional aspects related to regulation. Specifically, the example of the figure presents the station stop process, where a human sets the stop

mode and the system opens the platform passenger doors. The elements of Fig. 1 highlighted in gray conform an example model fragment.

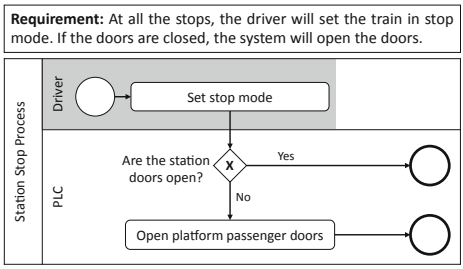

Fig. 1. Example of requirement, model and model fragment

2.2 Tacit Knowledge in Requirements

However, the requirement in Fig. 1 is lacking important information, known by the engineers and kept as tacit knowledge. A literal interpretation of the second sentence of the requirement implies that at all stations, all the doors of the train will open. However, the sentence embodies tacit knowledge that is not written but that is obvious to the domain engineers: (1) the train has doors on both sides, but only the doors on the side of the platform will open; and (2) not all the doors will open, the door of the control cabin will remain closed for the safety of the driver and the train. Thus, only the platform passenger doors will open.

3 Our Approach

3.1 Approach Overview

Through the presented approach, we tackle the tacit knowledge issue presented in the prior section, by expanding requirements through a domain ontology. The approach runs in a two-step process:

1. First, we use Natural Language Processing (NLP) techniques to process the requirement and the ontology. The NLP techniques unify the language of the software artifacts, which facilitates the expansion process.
2. Secondly, we propose an Ontological Requirement Expansion (ORE) process that uses the processed requirement and ontology in order to expand the requirement with related domain knowledge, diminishing the amount of tacit knowledge in the requirement.

The expanded requirement is used along with the NLP-treated process models from our case study as an input for Latent Semantic Indexing (LSI) [12], a widely accepted TLR process [26]. Through LSI, a model fragment, candidate solution for the requirement, is produced. Figure 2 depicts an overview of the steps of the approach. In the figure, rounded boxes represent the inputs and outputs of each step, while squared boxes represent each step. The highlighted boxes represent the initially available inputs (requirement, ontology, and model) used for the different steps of our approach and for the TLR process, and the final output (the most relevant model fragment for the requirement).

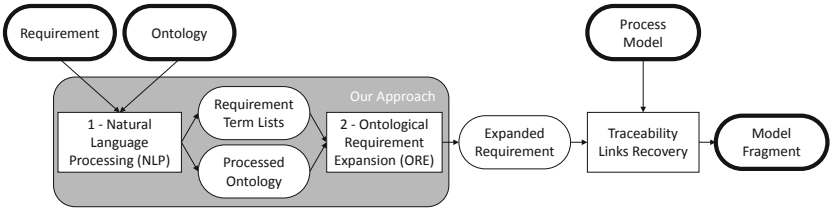

Fig. 2. Approach overview

3.2 Natural Language Processing (NLP)

This section describes the NLP techniques taken in account for our approach. Figure 3 is used to illustrate the whole compendium of techniques, detailed through the following paragraphs.

Splitting: As seen in Sect. 2, the tacit knowledge lies within the sentences of the requirements. Thus, in order to better isolate the tacit knowledge issue, we split the text of the requirements into the sentences that compose it. These smaller parts of text will help expand the requirement more accurately further on in our approach. Figure 3 depicts the two sentences that result from splitting the running example requirement.

Syntactical Analysis: Syntactical Analysis (SA) techniques analyze the specific roles of each one of them in the sentence and determine their grammatical function. These techniques (referred to as Parts-Of-Speech Tagging, or POS Tagging) allow engineers to implement filters for words that fulfill specific grammatical roles in a requirement, usually opting only for nouns [5]. In Fig. 3, it is possible to appreciate the SA process, with the POS Tagged Tokens associated to each sentence of the requirement as outcome.

Root Reduction: The technique known as Lemmatizing reduces words to their semantic roots or lemmas. Thanks to lemmas, the language of the NL requirements is unified, avoiding verb tenses, noun plurals, and other word forms that interfere negatively with the TLR process. The unification of the language semantics is an evolution over pure syntactical role filtering, allowing for a more advanced filtering of words in NL requirements. In Fig. 3, it is possible to

appreciate the RR process, with the Root-Reduced Tokens as outcome of the semantic analysis of the POS Tags derived from the NL requirement (keeping only nouns). This process is also applied to the ontology, treating all the concepts as nouns, since domain terms always name important characteristics of the trains.

Human NLP: The inclusion of domain knowledge through experts and software engineers in the TLR process is regarded as beneficial. Human NLP is often carried out through Domain Terms Extraction or Stopwords Removal. In our approach, domain terms are checked for after splitting the requirement into sentences. We analyze each sentence in search for the domain terms provided by the software engineers, and add the found domain terms to the final processed sentence. On the other hand, stopwords are filtered out of the Root Reduced sentences. Figure 3 depicts the Human NLP process, where a software engineer provides both lists of terms, which are consequently introduced into the final query, or filtered out of it.

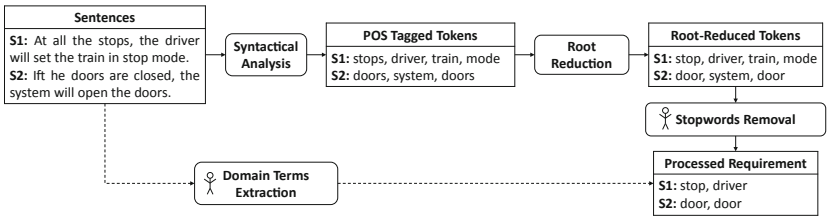

Fig. 3. Natural language processing techniques

3.3 Ontological Requirement Expansion (ORE)

The process that we propose in order to ontologically expand a requirement is detailed through the paragraphs of this section. The process runs in two steps: (1) calculation of the Ontological Affinity Documents associated to the requirement, and (2) expansion of the requirement.

1. **Ontological Affinity:** Ontological Affinity Documents (OADs) are documents that contain a set of ontological concepts related to a certain input. The first step of the Ontological Requirement Expansion process is to calculate the OADs associated to the requirement. We designed an algorithm that utilizes a processed domain ontology and a processed requirement to generate the OADs. The algorithm first selects one of the processed sentences generated through NLP. Then, the algorithm takes one term in the sentence, searching for it in the ontology. If the term matches a concept that is present in the ontology, all the concepts directly connected to the concept are added to an OAD. The algorithm iterates over all the terms in the sentence, generating the OAD associated to the sentence. The process is repeated for every

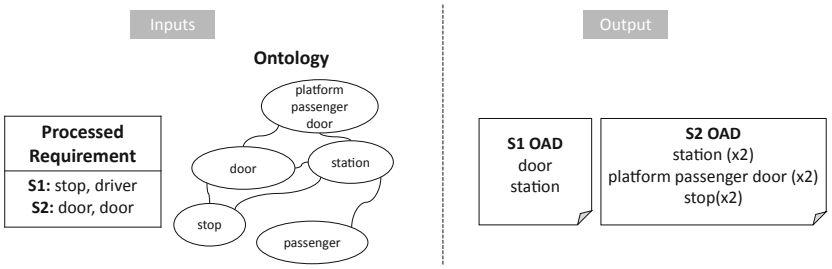

Fig. 4. Ontological affinity documents calculation

sentence in the processed requirement, generating one OAD per sentence. This process is illustrated through our running example in Fig. 4. In the figure, for space reasons, only a small part of the domain ontology is represented. In the case of the first sentence, the term 'stop' appears both in the sentence and as an ontology concept. The concepts that are directly related to the 'stop' concept are 'station' and 'door'. These concepts are therefore included into the OAD of the sentence. In our example, the term 'driver' does not appear as a concept in the ontology, providing no concepts for the OAD of the first sentence. In the case of the second sentence, the term 'door' appears as a concept in the example ontology. The concept is connected to 'station', 'stop', and 'platform passenger door'. Since the term appears twice in the sentence, the concepts are added twice to the OAD.

2. **Requirement Expansion:** Through this step, our approach automatically reformulates the processed requirement to expand it with terms of the OADs using a technique that is based on Rocchio's method [18], which is perhaps the most commonly used method for query reformulation [20]. Rocchio's method orders the terms in the OADs based on the sum of the importance of each term of the documents using the following equation:

$$Rocchio = \sum_{d \in R} TF(c, d) \cdot IDF(t, R) \tag{1}$$

Where R is the set of OADs, d is a document in R, and c is a concept in d. The first component of the measure is the Term Frequency (TF), which is the number of times the concept appears in a document; it is an indicator of the importance of the concept in the document compared to the rest of the concepts in that document. The second component is the Inverse Document Frequency (IDF), which is the inverse of the number of documents that contain that concept; it indicates the specificity of that concept for a document that contains it. The IDF measurement is calculated as:

$$IDF(t, R) = \log \frac{|R|}{|\{d \in R : c \in d\}|} \tag{2}$$

Where $|R|$ is the number of documents and $|\{d \in R : c \in d\}|$ is the number of documents where the concept is present.

To illustrate this calculation, consider the processed requirement from our running example. After calculating the OADs presented in Fig. 4, Rocchio's method is applied to the concepts of the documents in order to retrieve the importance of said concepts. Take in account the concept 'platform passenger door'. In the first document, the concept does not appear ($TF = 0$), immediately leading to a $TF \cdot IDF$ value of $TF \cdot IDF = 0$. The concept appears twice in the second document ($TF = 2$) and appears in one of two documents ($IDF = \log \frac{2}{1} \approx 0.3$), which leads to a $TF \cdot IDF$ value of $TF \cdot IDF \approx 0.6$. The sum of both $TF \cdot IDF$ values leads to a total *Rocchio* value of $Rocchio \approx 0.6$. Using Rocchio's method, the concepts of the OADs associated to the sentences of the requirement are ordered from highest to lowest sum of importance into a single document of concepts. Once ordered, we take in consideration only the first 10 suggestions and discard the rest, as is recommended in the literature [6]. The list of the 10 first suggested concepts conforms the OAD associated to the requirement.

Since the objective of our approach is to mitigate the tacit knowledge of the requirement, our aim is to find new domain knowledge to include in the requirement, and therefore we refine the requirement OAD by discarding those concepts in the OAD that already appear in any sentence of the requirement. In our running example, this process would produce a requirement OAD consisting of the terms 'station' and 'platform passenger door', since both 'door' and 'stop' are already present in the sentences of the requirement. The terms of the processed sentences and the concepts on the refined OAD are then concatenated into a single list of terms. This final list of terms is the ultimate goal that our approach seeks to obtain: an expanded requirement, enriched with ontological domain knowledge. The expanded requirement is the final output of the Ontological Requirement Expansion process, and is used as query for the Traceability Links Recovery process.

4 Traceability Links Recovery

LORE can be applied to any TLR technique that uses a requirement as input. Through this work, we utilize Latent Semantic Indexing (LSI), the TLR technique that obtains the best results when performing TLR between requirements and software artifacts [26]. Latent Semantic Indexing (LSI) [12] constructs vector representations of a query and a corpus of text documents by encoding them as a *term-by-document co-occurrence matrix*. In our approach, *terms* are each of the words that compose the expanded requirement and NL representation of the input model (extracted through the technique presented in [15]), *documents* are the model elements in the input model, and the *query* is the expanded requirement. Each cell in the matrix contains the frequency with which the *term* of its row appears in the *document* denoted by its column. Once the matrix is built, it is normalized and decomposed into a set of vectors using a matrix factorization technique called Singular Value Decomposition (SVD) [12].

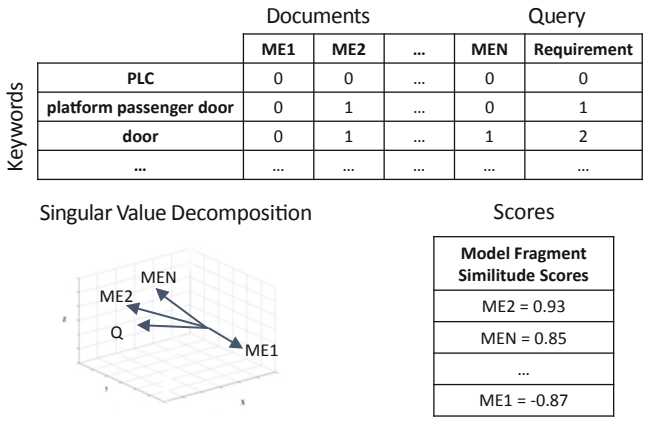

Fig. 5. Traceability link recovery through latent semantic indexing example

The similarity degree between the query and each document is calculated through the cosine between the vectors that represent them. Figure 5 shows an example matrix, built from our running example, the result of applying the SVD technique to the matrix, and the resulting scores associated to each document. In our approach, we use the top ranked model elements to build a model fragment that serves as a candidate for realizing the requirement. Of all the model elements, only those that have a similarity measure greater than x must be taken into account. A widely used heuristic is $x = 0.7$. This value corresponds to a 45° angle between the corresponding vectors. Even though the selection of the threshold is an issue under study, the chosen heuristic has yielded good results in other similar works [14,17].

5 Evaluation

This section presents the evaluation of our approach, including the experimental setup, a description of the case study where we applied the evaluation, and the implementation details of our approach.

5.1 Experimental Setup

The goal of this experiment is to perform TLR between requirements and models through LORE, comparing its results against the baseline. The baseline against which we compare our work is the technique that obtains the best results when recovering Traceability between requirements and models according to the literature, TLR through LSI. The baseline utilizes the processed requirement, without performing the ontological expansion in use in LORE. Figure 6 shows an overview of the process followed to evaluate our approach (LORE) and the baseline (TLR). The top part of the figure shows the inputs, as provided by our industrial partner. The requirements and models are used to build the test cases

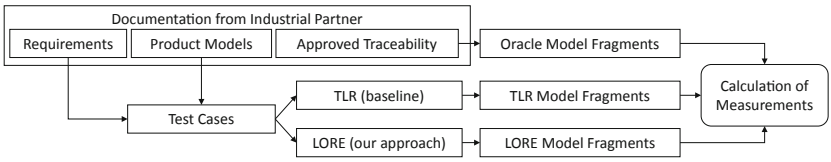

Fig. 6. Experimental setup

(one requirement and one model each) and the approved Traceability is used to build the oracles against which the results of the approaches are compared.

For each test case, both LORE and TLR generate one model fragment each. The model fragments generated for each test case are compared against their respective oracles (ground truth), and a confusion matrix is calculated for each of the two approaches. A confusion matrix is a table used to describe the performance of a classification model on a set of test data for which the true values are known. In our case, the presence or absence of each model element is considered as a classification. The confusion matrix arranges the results of the classifications into four categories: (1) True Positive (predicted true, true in the real scenario), (2) False Positive (predicted true, false in the real scenario), (3) True Negative (predicted false, false in the real scenario), and (4) False Negative (predicted false, true in the real scenario). From the confusion matrix, it is possible to extract some measurements that evaluate the performance of the approach. We report four performance measurements for both LORE and TLR: Recall, Precision, F-measure, and MCC (Matthews Correlation Coefficient). Recall measures the number of elements of the solution that are correctly retrieved by the proposed solution, precision measures the number of elements from the solution that are correct according to the ground truth, and the F-measure corresponds to the harmonic mean of precision and recall. The MCC is a correlation coefficient between the observed and predicted binary classifications [14,19].

Recall values can range between 0% (no single model element from the oracle is present in the retrieved model fragment) to 100% (all the model elements from the oracle are present in the retrieved model fragment). Precision values can range between 0% (no model elements from the retrieved model fragment appear in the oracle) to 100% (all the model elements from the retrieved model fragment appear in the oracle). MCC values can range between -1 (no correlation between the prediction and the oracle) to 1 (perfect prediction). Moreover, an MCC value of 0 corresponds to a random prediction.

5.2 Case Study

The case study where we applied our approach was provided by our industrial partner, CAF (http://www.caf.es/en), a worldwide provider of railway solutions. Our evaluation includes 140 test cases, with each test case comprising one requirement, one model, and the approved Traceability between the requirement and the model. The requirements have about 25 words on average, and

the models are formed through 650 elements on average. For each test case, we followed the experimental setup described in Fig. 6.

Regarding the domain ontology in use, it comprises 27 concepts and 176 relationships. The construction of an ontology is a major effort which requires the study of the domain structure and terminology. We did not try to address the creation of a new ontology in this paper but instead, our industrial partner provided us with the ontology they use for training new employees. The ontology is an important artifact and that its quality, size, and completeness may have an impact on the results. In a future work, we intend to analyze the extent of this impact on the results of LORE.

5.3 Implementation Details

For the development of the Natural Language Processing operations used in both our approach and the baseline, we have used the OpenNLP Toolkit [2]. To implement the LSI and SVD techniques, the Efficient Java Matrix Library (EJML) was used [1]. For the evaluation, we used a Lenovo E330 laptop, with a processor Intel(R) Core(TM) i5-3210M@2.5 GHz with 16 GB RAM and Windows 10 (64-bit). A prototype of LORE can be found at bitbucket.org/svitusj/lore.

6 Results

Table 1 outlines the results of the TLR baseline and our LORE approach. Each row shows the Precision, Recall, F-measure, and MCC values obtained through each of the two approaches. The LORE approach achieves the best results for all the performance indicators, providing a mean precision value of 79.2%, a mean recall value of 50.2%, a combined F-measure of 66.5%, and an MCC value of 0.62. In contrast, the TLR baseline presents worse results in all the measurements, attaining a mean precision value of 59.3%, a mean recall value of 45.5%, a combined F-measure of 52.4%, and an MCC value of 0.31. We also included the values of the measurements for the top 20 and the bottom 20 results for TLR and LORE, to better highlight how the results obtained by LORE improve those obtained by the TLR baseline.

Table 1. Mean values and standard deviations for precision, recall and F-measure

	Precision	Recall	F-measure	MCC
TLR	$59.3\% \pm 29.6\%$	$45.5\% \pm 34.2\%$	$52.4\% \pm 31.9\%$	0.31 ± 0.13
LORE	$79.2\% \pm 33.6\%$	$50.2\% \pm 30.6\%$	$66.5\% \pm 38.6\%$	0.62 ± 0.32
Top 20 - TLR	$81.3\% \pm 7.3\%$	$55.4\% \pm 3.2\%$	$68.3\% \pm 5.2\%$	0.41 ± 0.03
Top 20 - LORE	$93.4\% \pm 8.4\%$	$69.8\% \pm 4.6\%$	$81.6\% \pm 6.5\%$	0.86 ± 0.04
Bottom 20 - TLR	$48.3\% \pm 6.9\%$	$19.8\% \pm 4.2\%$	$34.1\% \pm 5.5\%$	0.22 ± 0.04
Bottom 20 - LORE	$66.2\% \pm 5.7\%$	$41.2\% \pm 5.1\%$	$53.7\% \pm 5.4\%$	0.38 ± 0.08

7 Discussion

The results presented in the previous section suggest that by embedding domain knowledge into requirements the TLR process retrieves enhanced results. Taking a closer look at the test cases, we found out that there are many terms in the models that do not appear in the requirements. Through the ontological expansion of the requirements, they are enriched with otherwise missing terms, retrieving more and better links. However, we also noticed a series of facts that prevent LORE from achieving better results than it does. We should tackle these issues in the future to further improve our line of work:

1. Our analysis of the results raised awareness about the importance of the quality and completeness of the ontology in LORE. If a particular concept does not have quality connections, the quality of the expansion process is diminished, also affecting the quality of the final outcome. Equally, if a concept is missing from the ontology, the concept itself and its would-be related concepts cannot be introduced in the expanded requirement. This issue leaves parts of the domain knowledge out from the requirement, causing a decrease in recall. In order to tackle this issue, we plan to automatically identify words and patterns of words that occur repeatedly in the requirements and models, and suggest their inclusion in the ontology as concepts, entrusting the creation of their relationships to the software engineers.
2. In the ontology, we identified some terms that have a large number of connections to other terms. Matching one of those terms through LORE leads to the inclusion of several unwanted ontological concepts into the expanded requirement. This concatenation of events reflects into LSI noise, strongly affecting in a negative manner the precision of the results, since elements that are not part of the oracle can be added to the proposed solution due to this issue. To tackle this issue, we plan to automatically identify the overly connected ontological concepts and suggest their inclusion in the stopwords list to the software engineers, so they can be kept out of the LORE analysis.
3. Another possible consideration towards the obtained results is the parameter tuning of our approach. Many Information Retrieval approaches have parameters that can be tuned in order to improve the results (such as the LSI similitude threshold), and our approach is no exception. So far, we have considered only the directly related ontological concepts when performing the expansion (one jump or ontological affinity level 1). In the future, we plan to study how using different levels of affinity may impact the results. We believe this could help us further explore the ontology and the relationships between the concepts, although at a risk of including noise into LSI. Analyzing the tuning of this parameter and its implications and impact on the outcomes of the LORE approach remains as future work.
4. Regarding recall, we have inspected the results and have determined that the low recall levels are not dependent solely on the techniques under use, but are also affected by the quality of the received queries, which in several occasions, are poorly formulated. Focusing only on these particular cases, recall values

obtained by TLR range at 20%, while recall values obtained by LORE range at 40%. However, for better quality queries, TLR recall results range at 55%, while those of LORE range at 70%. The point is, considering ontological knowledge in the process helps improve traceability results. That is, in the face of poor quality inputs, the results improve, but if we feed LORE with better queries, the results improve as well. Studying the quality of the inputs and how to ensure it remains as an interesting research topic for a future work in which we might as well design another experiment to research how LORE improves the results of TLR for top-quality queries. In any case, as a naive experiment and in order to ensure the usefulness of the obtained results, we have discussed them with one of the software engineers working for our industrial partner, who has confirmed that the model fragments obtained by LORE serve as a better starting point for requirement-model tracing than those obtained by plain TLR.

5. Finally, in many cases, different terms are used to reference the same concept in the requirements, models, and ontology alike. In industrial environments, the engineers in charge of writing requirements may not be assigned with the building of the models or the ontology in any ways, being those tasks left for different engineers. Moreover, the artifacts can be manipulated by different engineers. This issue is known as vocabulary mismatch. Even though LORE uses NLP to homogenize the language between requirements and models, the vocabulary mismatch continues to be a disregarded issue in our work. The lack of awareness caused by the vocabulary mismatch makes it impossible to locate the elements from the model that are relevant to the requirement, which in turn negatively impacts both precision and recall. To mitigate this issue, we plan on adding a third human-made list, comprising in-house terms and their possible synonyms, allowing us to further map ontology concepts and requirements.

8 Threats to Validity

In this section, we use the classification of threats to validity of [27] to acknowledge the limitations of our approach.

1. **Construct Validity:** This aspect of validity reflects the extent to which the operational measures that are studied represent what the researchers have in mind. To minimize this risk, our evaluation studies four measures that are widely accepted in the software engineering research community: precision, recall, F-measure, and MCC.
2. **Internal Validity:** This aspect of validity is of concern when causal relations are examined. There is a risk that the factor being investigated may be affected by other neglected factors. The number of requirements and models presented in this work may look small, but they implement a wide scope of different railway equipment.
3. **External Validity:** This aspect of validity is concerned with to what extent it is possible to generalize the findings, and to what extent the findings are

of relevance for other cases. Both requirements and process models are frequently leveraged to specify all kinds of different software. LSI is a widely accepted and utilized technique which has proven to obtain good results in multiple domains. The NLP techniques studied through this work are also commonly used in the whole of the SE community. Therefore, our experiment does not rely on the particular conditions of our domain. In addition, the real-world models used in our research are a good representative of the railway, automotive, aviation, and general industrial manufacturing domains. Nevertheless, the experiment and its results should be replicated in other domains before assuring their generalization.

4. **Reliability:** This aspect is concerned with to what extent the data and the analysis are dependent on the specific researchers. To reduce this threat, all the software artifacts were provided by our industrial partner.

9 Related Work

Some works focus on the impact and application of Linguistics to TLR at several levels of abstraction. Works like [23,24] or [7] use Linguistic approaches to tackle specific TLR problems and tasks. In [9], the authors use Linguistic techniques to identify equivalence between requirements, also defining and using a series of principles for evaluating their performance when identifying equivalent requirements. The authors of [9] conclude that, in their field, the performance of Linguistic techniques is determined by the properties of the given dataset over which they are performed. They measure the properties as a factor to adjust the Linguistic techniques accordingly, and then apply their principles to an industrial case study. The work presented in [4] uses Linguistic techniques to study how changes in requirements impact other requirements in the same specification. Through the pages of their work, the authors analyze TLR between requirements, and use Linguistic techniques to determine how changes in requirements must propagate.

Our work differs from [23,24] and [7] since our approach is not based on Linguistic techniques as a means of TLR, but we rather use an ontological expansion process to enrich requirements before performing TLR, using NLP techniques only as a preprocess in our work. Moreover, we do not study how Linguistic techniques must be tweaked for specific problems as [9] does. In addition, differing from [4], we do not tackle changes in requirements nor TLR between requirements, but instead focus our work on TLR between requirements and models.

Other works target the application of LSI to TLR tasks. De Lucia et al. [13] present a Traceability Links Recovery method and tool based on LSI in the context of an artifact management system, which includes models. [8] takes in consideration the possible configurations of LSI when using the technique for TLR between requirements artifacts, namely requirements and test cases. In their work, the authors state that the configurations of LSI depend on the datasets used, and they look forward to automatically determining an appropriate configuration for LSI for any given dataset. Through our work, we do not

focus on the usage of LSI or its tuning, but rather expand requirements with ontological domain knowledge before carrying out TLR between said requirements and the models.

10 Conclusions

Through this work, we propose a novel approach (LORE), based on an Ontological Requirement Expansion process, that can be used to minimize the impact that tacit knowledge has on TLR. We evaluated our approach by carrying out LORE between the requirements and process models that comprise a real-world industrial case study. Results show that our approach guides TLR to the best results for all the measured performance indicators, providing a mean precision value of 79.2%, a mean recall value of 50.2%, a combined F-measure of 66.5%, and an MCC value of 0.62. In contrast, the baseline used for comparison presents worse results in these same measurements. In addition, we identified a series of issues that prevent our approach from achieving better solutions, and that should be tackled in the future in order to further improve the TLR process between requirements and process models. To facilitate the adoption of LORE, we made a reference implementation freely available for the Eclipse environment.

Acknowledgements. This work has been partially supported by the Ministry of Economy and Competitiveness and ERDF funds under the project *Model-Driven Variability Extraction for Software Product Lines Adoption* (TIN2015-64397-R). We also thank the ITEA3 15010 REVaMP2 Project.

References

1. Abeles, P.: Efficient Java Matrix Library (2017). http://ejml.org/. Accessed 9 Nov 2017
2. Apache: OpenNLP Toolkit for the Processing of Natural Language Text (2017). https://opennlp.apache.org/. Accessed 12 Nov 2017
3. Arora, C., Sabetzadeh, M., Briand, L., Zimmer, F.: Extracting domain models from natural-language requirements: approach and industrial evaluation. In: Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems, pp. 250–260. ACM (2016)
4. Arora, C., Sabetzadeh, M., Goknil, A., Briand, L.C., Zimmer, F.: Change impact analysis for natural language requirements: an NLP approach. In: IEEE 23rd International Requirements Engineering Conference (2015)
5. Capobianco, G., De Lucia, A., Oliveto, R., Panichella, A., Panichella, S.: On the role of the nouns in IR-based traceability recovery. In: IEEE 17th International Conference on Program Comprehension, ICPC 2009, pp. 148–157. IEEE (2009)
6. Carpineto, C., Romano, G.: A survey of automatic query expansion in information retrieval. ACM Comput. Surv. **44**, 1:1–1:50 (2012)
7. Duan, C., Cleland-Huang, J.: Clustering support for automated tracing. In: Proceedings of the 22nd IEEE/ACM International Conference on Automated Software Engineering (2007)

8. Eder, S., Femmer, H., Hauptmann, B., Junker, M.: Configuring latent semantic indexing for requirements tracing. In: Proceedings of the 2nd International Workshop on Requirements Engineering and Testing (2015)
9. Falessi, D., Cantone, G., Canfora, G.: Empirical principles and an industrial case study in retrieving equivalent requirements via natural language processing techniques. Trans. Softw. Eng. **39**(1), 18–44 (2013)
10. Ghazarian, A.: A research agenda for software reliability. IEEE Reliability Society 2009 Annual Technology Report (2010)
11. Gotel, O.C., Finkelstein, C.: An analysis of the requirements traceability problem. In: Proceedings of the First International Conference on Requirements Engineering, pp. 94–101. IEEE (1994)
12. Landauer, T.K., Foltz, P.W., Laham, D.: An introduction to latent semantic analysis. Discourse Process. **25**(2–3), 259–284 (1998)
13. de Lucia, A., et al.: Enhancing an artefact management system with traceability recovery features. In: Proceedings of the 20th IEEE International Conference on Software Maintenance, pp. 306–315. IEEE (2004)
14. Marcus, A., Sergeyev, A., Rajlich, V., Maletic, J.: An information retrieval approach to concept location in source code. In: Proceedings of the 11th Working Conference on Reverse Engineering, pp. 214–223 (2004). https://doi.org/10.1109/WCRE.2004.10
15. Meziane, F., Athanasakis, N., Ananiadou, S.: Generating natural language specifications from UML class diagrams. Requirements Eng. **13**(1), 1–18 (2008)
16. Rempel, P., Mäder, P.: Preventing defects: the impact of requirements traceability completeness on software quality. IEEE Trans. Softw. Eng. **43**(8), 777–797 (2017)
17. Salman, H.E., Seriai, A., Dony, C.: Feature location in a collection of product variants: combining information retrieval and hierarchical clustering. In: The 26th International Conference on Software Engineering and Knowledge Engineering, pp. 426–430 (2014)
18. Salton, G.: The SMART Retrieval System - Experiments in Automatic Document Processing. Prentice-Hall Inc., Upper Saddle River (1971)
19. Salton, G., McGill, M.J.: Introduction to Modern Information Retrieval. McGraw-Hill Inc., New York (1986)
20. Sisman, B., Kak, A.C.: Assisting code search with automatic query reformulation for bug localization. In: Proceedings of the 10th Working Conference on Mining Software Repositories, pp. 309–318 (2013)
21. Spanoudakis, G., Zisman, A.: Software traceability: a roadmap. Handb. Softw. Eng. Knowl. Eng. **3**, 395–428 (2005)
22. Stone, A., Sawyer, P.: Using pre-requirements tracing to investigate requirements based on tacit knowledge. In: ICSOFT (1), pp. 139–144 (2006)
23. Sultanov, H., Hayes, J.H.: Application of swarm techniques to requirements engineering: requirements tracing. In: 18th IEEE International Requirements Engineering Conference (2010)
24. Sundaram, S.K., Hayes, J.H., Dekhtyar, A., Holbrook, E.A.: Assessing traceability of software engineering artifacts. Requirements Eng. **15**(3), 313–335 (2010)
25. Watkins, R., Neal, M.: Why and how of requirements tracing. IEEE Softw. **11**(4), 104–106 (1994)
26. Winkler, S., Pilgrim, J.: A survey of traceability in requirements engineering and model-driven development. Softw. Syst. Model. (SoSyM) **9**(4), 529–565 (2010)
27. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29044-2