On the Influence of Models-to-Natural-Language Transformation in Traceability Link Recovery among Requirements and Conceptual Models

Raúl Lapeña, Francisca Pérez, and Carlos Cetina

SVIT Research Group, Universidad San Jorge
Autovía A-23 Zaragoza-Huesca Km.299
50830 Villanueva de Gállego (Zaragoza), Spain
{rlapena,mfperez,ccetina}@usj.es

Abstract. Recovering traceability links between software artifacts and requirements is a common task in Software Engineering. Information Retrieval (IR) techniques have been applied to recover traceability links amongst code and requirements. By transforming Models into Natural Language (M2NL), it is possible to apply IR to calculate their traceability links to requirements. However, results retrieved by IR are affected by the writing style of the NL input. Regarding M2NL, there are two main types of techniques in use: Rule-Based techniques, and Element-Based techniques. Along with M2NL, there is a wide range of Natural Language Processing (NLP) techniques that can be applied. Through this work, we analyze how the usage of distinct M2NL-NLP combinations of techniques impacts IR-based Traceability Links Recovery over requirements and models. We evaluate two different M2NL techniques, and the inclusion of Simple and Advanced NLP along with M2NL, in a real-world industrial case study.

Keywords: Natural Language Processing, Traceability Link Recovery, Domain Specific Language

1 Introduction

Traceability Links Recovery (TLR) between software artifacts and requirements is a common task in Software Engineering (SE), specially when maintaining and evolving software products. For code and Natural Language (NL) requirements, Information Retrieval (IR) techniques have been successfully used for TLR. By transforming conceptual models into NL, it is possible to apply IR to requirements-models TLR. However, results retrieved by these techniques depend greatly in the style in which NL is written. Two main types of techniques are applied for Models-to-Natural-Language Transformation (M2NL): Rule-Based techniques [1], and Element-Based techniques [2]. Rule-Based techniques apply sets of logical and grammatical rules, while Element-Based techniques extract text associated to model elements directly.

After the M2NL transformation process, there is a wide range of Natural Language Processing (NLP) techniques that are applied to process NL representations of models. Some of them are: general phrase styling techniques, syntactical analysis techniques [3], semantic analysis techniques [4], and human-in-the-loop techniques. These techniques are combined in distinct ways by different authors, depending on implementation circumstances and research particularities.

The impact of the usage of different M2NL-NLP techniques combinations on requirements-models TLR has not been studied yet. Through this work, we analyze how distinct M2NL-NLP techniques combinations impact requirements-models TLR through Latent Semantic Indexing (LSI) [5], the technique that obtains the best TLR results [6]. We evaluate two different M2NL techniques and the inclusion of Simple and Advanced NLP along with M2NL, in a real-world industrial case study in the rolling stocks domain with our industrial partner, Construcciones y Auxiliar de Ferrocarriles (CAF, `http://www.caf.net/en`).

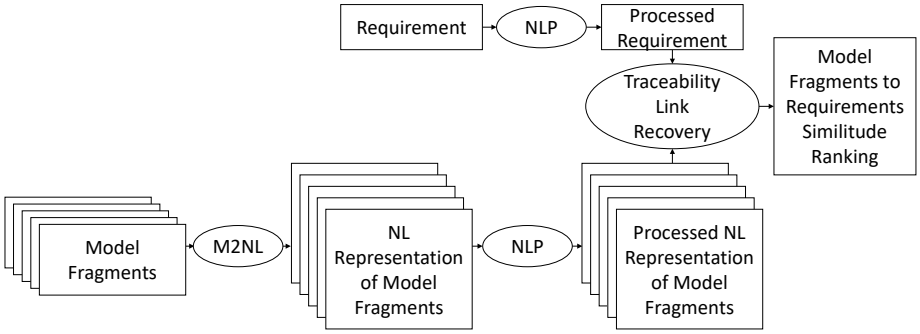
The combination of Rule-Based M2NL with Advanced NLP leads LSI to the best results, returning the model fragments that materialize requirements in an average ranking position of 1 ± 1.12. However, in order to use Rule-Based M2NL, engineers must adapt or create rules for their Domain Specific Language (DSL). The combination of Element-Based M2NL with Advanced NLP returns a worse result for the same measurement (2 ± 5.09), but does not require said efforts.

The paper is structured as follows: Section 2 presents our Approach. Section 3 details the Evaluation designed to tackle the Research Questions. Section 4 analyzes the statistical significance of the obtained results. Section 5 presents the Threats to Validity of our work. Section 6 summarizes the works related to the presented paper. Finally, Section 7 concludes the paper.

2 Approach

So far, there has been no discussion on which Model-to-Natural-Language Transformation (M2NL) techniques should be applied for requirements-models Traceability Links Recovery (TLR). The effect of the inclusion of Simple or Advanced Natural Language Processing (NLP) techniques along with M2NL to the same intent has not been studied yet either. The presented approach studies the impact of using two different M2NL techniques, and the impact of including Simple or Advanced NLP techniques along M2NL, over a widely accepted TLR technique, Latent Semantic Indexing (LSI). Analyzing the success of LSI over the different inputs, we aim to determine which one guides LSI to enhanced results.

The top part of Fig. 1 depicts the outline of this work. Through the usage of M2NL techniques, we convert model fragments into NL. Then, we process the NL representation of the models and a NL requirement from our case study through NLP techniques. With the processed model fragments and requirement, we carry out LSI, ranking the model fragments according to their similitude to the query requirement. The bottom part of Fig. 1 shows the four configurations considered through this work, which we analyze in order to determine their impact on requirements-models TLR.



	Configuration 1	Configuration 2	Configuration 3	Configuration 4
M2NL	Element-Based	Element-Based	Rule-Based	Rule-Based
NLP	Simple	Advanced	Simple	Advanced

Fig. 1. Traceability Link Recovery among Requirements and Conceptual Models Overview

The following subsections describe the M2NL techniques taken in account through the rest of this work, the NLP techniques used to process the NL representations of the model fragments and requirements in the case study, and the LSI technique from which results are extracted.

2.1 Models-to-Natural-Language Transformation Techniques (M2NL)

In order to extract NL from models, two main techniques are applied in the literature: Rule-Based, and Element-Based M2NL. Fig. 2 depicts an example DSL model from our industrial partner (where the company-specific DSL in use is TCML, Train Control Modeling Language), and shows the results of applying both Rule-Based and Element-Based M2NL to the model. In order to formalize model fragments, we use the Common Variability Language (CVL) [7].

2.1.1 Rule-Based M2NL

This technique uses a set of user-defined rules to process text inside models. Through the rules, several aspects inherent to modeling language (such as naming conventions, model element types, grammatical element ordering, etc.) are exploited to generate semantically sound NL representations of models.

We use the Rule-Based M2NL technique presented by Meziane et. al. in [1], where the authors research the language used in class diagrams components and develop rules for semantically sound NL generation. The TCML used by our industrial partner was developed following UML conventions, with equipment elements corresponding to UML classes, equipment properties corresponding to UML attributes, and connections corresponding to UML relationships. We can

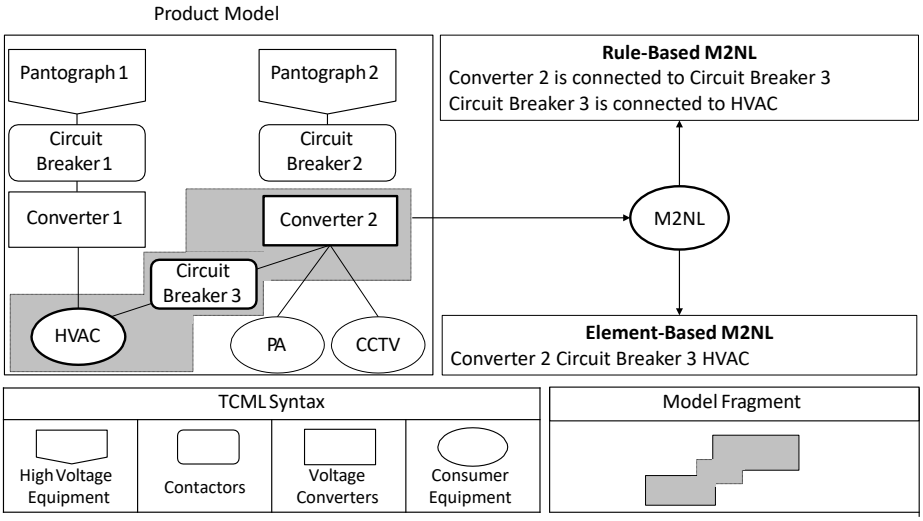


Fig. 2. Example of TCML model and model fragment

leverage the rules in [1] to generate NL representations of the TCML models. As an example, for the model fragment depicted in the top left part of Fig. 2, this technique would yield the following strings: 'Converter 2 is connected to Circuit Breaker 3', and 'Circuit Breaker 3 is connected to HVAC'.

2.1.2 Element-Based M2NL

This technique is used in approaches that concur M2NL for Feature Location [2] and Software Product Lines synthesis [8] purposes. Through this technique, the NL texts that represent each element of a model are extracted and then concatenated into a single string, used as a NL representation of the model. The NL representations generated through this technique vary in understandability, being commonly closer to a collection of words or expressions than to descriptions of functionality understandable by humans.

For the TCML models from our industrial partner, we use this technique by extracting the text from all the model elements. As an example, for the model fragment depicted in the top left part of Fig. 2, this technique would yield the string 'Converter 2 Circuit Breaker HVAC'. Fig. 2 is, for understandability purposes and space reasons, a simplification of a real model. In a real model, model elements contain more properties which in turn yield more text in its NL representation.

2.2 NLP Techniques

Fig. 3 depicts the NLP techniques used through this work, along with an example taken from a real-world train. In Fig. 3, a NL requirement is used as the input for the example, but through our work, these techniques are applied to both NL requirements and NL representations of models obtained through the application of either Rule-Based M2NL or Element-Based M2NL (see top part of Fig. 1).

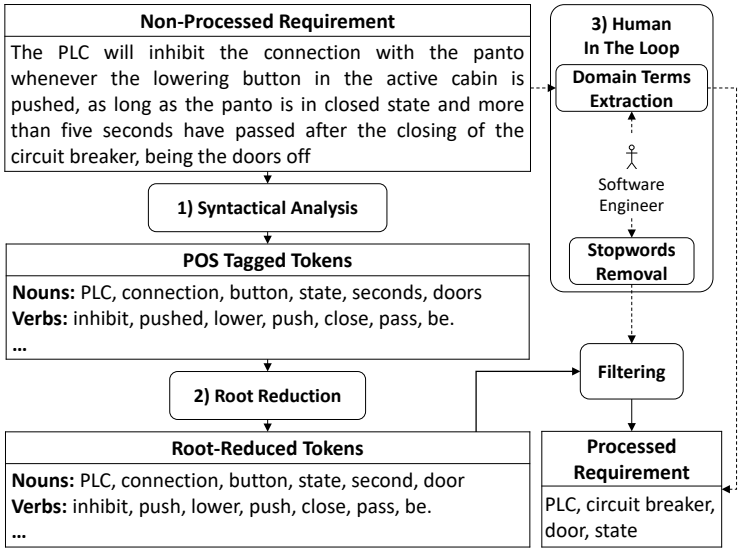


Fig. 3. Compendium of NLP Techniques

Through this work, we include either Simple or Advanced NLP along with M2NL. Simple NLP uses the techniques in 2.2.4, since we consider their combination to be the most basic unit of NLP. Advanced NLP includes the techniques that conform Simple NLP, plus those described in 2.2.1, 2.2.2, and 2.2.3.

2.2.1 Syntactical Analysis

Syntactical Analysis (SA) techniques determine the grammatical function of words in sentences (e.g.: nouns, verbs, etc.). These techniques, often referred to as Parts-Of-Speech (POS) Tagging, allow engineers to implement grammatical filters, usually in search for nouns, which often carry relevant information on features and actions [9]. Words like verbs or adjectives are often disregarded. In Fig. 3, it is possible to appreciate the SA process, with the POS Tagged Tokens as outcome of syntactically analyzing a real-world NL requirement. Nouns and verbs are depicted while, for space reasons, the rest of the words are omitted.

2.2.2 Root Reduction

Through the usage of semantic techniques such as Lemmatizing, words can be reduced to their semantic roots (lemmas). Through lemmas, it is possible to unify NL, avoiding verb tenses, plurals, and strange word forms that interfere with TLR. Prior to carrying out Root Reduction (RR) techniques, it is imperative to use SA techniques, since RR techniques are based on word dictionaries built upon the grammatical role of words. Semantic techniques provide more advanced word filters 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. The lemmas of nouns and verbs are depicted while, for space reasons, the rest of the words are omitted.

2.2.3 Human-In-The-Loop

The inclusion of domain experts in TLR processes is a widely discussed topic within SE. It is often beneficial to have domain knowledge embedded in TLR, particularly for software reuse and variability. Some of the human interaction techniques used in TLR are Domain Terms Extraction and Stopwords Removal. In order to carry out these techniques, engineers provide two separate lists of terms: one list of both single-word and multiple-word terms that belong to the domain and must be kept for analysis, and a list of irrelevant words that have no analysis value. Both kinds of terms can be automatically filtered in or out of the final query. In Fig. 3, it is possible to appreciate the Human-In-The-Loop process, where a software engineer provides both lists of terms, which are consequently introduced into the final query, or filtered out of it.

2.2.4 Other Filters

The most basic NLP technique covered in this work is the combination of tokenizing and lowercasing a sentence, and afterwards removing duplicate words from it. This combination is often regarded as the most basic NLP technique for several LSI examples.

2.3 Traceability Link Recovery through Latent Semantic Indexing

Latent Semantic Indexing (LSI) is an automatic mathematical/statistical technique that analyzes relationships between *queries* and *documents* (bodies of text). It constructs vector representations of both a user *query* and a corpus of text *documents* by encoding them as a *term-by-document co-occurrence matrix*, and analyzes the relationships between those vectors to get a similarity ranking between the *query* and the *documents*. Fig. 4 shows an example *term-by-document co-occurrence matrix*, with values associated to our case study, the vectors, and the resulting ranking. In the following paragraphs, an overview of the elements of the matrix is provided.

Terms: Each row in the matrix (*term*) stands for each of the words that compose the processed requirement and NL representations of model fragments. In Fig. 4, it is possible to appreciate a set of representative words in the domain such as 'pantograph' or 'doors' as the *terms* of each row.

Documents: Each column in the matrix stands for the processed NL representation of each model fragment in our case study. In Fig. 4, it is possible to appreciate the identifiers of the model fragments in the columns such as 'M_KAO001' or 'M_CIN072', which stand for the processed NL representations of those particular model fragments.

Query: The final column stands for the *query*. In our approach, the *query* is one processed requirement in our case study. In Fig. 4, the identifier of the requirement in the *query* column ('R_BUD010') represents its processed text.

Data: Each cell in the matrix contains the frequency with which the *term* of its row appears in the *document* denoted by its column. For instance, in Fig. 4, the *term* 'pantograph' appears twice in the 'M_KAO001' processed NL representation and once in the 'R_BUD010' processed requirement.

We obtain vector representations of the *documents* and the *query* by normalizing and decomposing the *term-by-document co-occurrence matrix* using a matrix factorization technique called *Singular Value Decomposition* (SVD) [5]. SVD is a form of factor analysis, or more properly the mathematical generalization of which factor analysis is a special case. In SVD, a rectangular matrix is decomposed into the product of three other matrices. One component matrix describes the original row entities as vectors of derived orthogonal factor values, another describes the original column entities in the same way, and the third is a diagonal matrix containing scaling values such that when the three components are matrix-multiplied, the original matrix is reconstructed.

In Fig. 4, a three-dimensional graph of the SVD is provided. On the graph, it is possible to appreciate the vectorial representations of some of the matrix columns. For space reasons, only a small set of the columns is represented. To measure the similarity degree between vectors, our approach calculates the cosine between the *query* vector and the *documents* vectors. Cosine values closer to one denote a higher degree of similarity, and cosine values closer to minus one denote a lower degree of similarity. Similarity increases as vectors point in the same general direction (as more *terms* are shared between *documents*). Through this measurement, our approach orders the model fragments according to their similarity degree to the requirement.

The relevancy ranking (which can be seen in Fig. 4) is produced according to the calculated similarity degrees. In this example, LSI retrieves 'M_BUD010' and 'M_KAO001' in the first and second position of the relevancy ranking due to *query-documents* cosines being '0.9243' and '0.8454', implying a high similarity degree between the fragments and the requirement. On the opposite, the 'M_CIN072' is returned in a latter position of the ranking due to its *query-document* cosine being '-0.7836', implying a lower similarity degree.

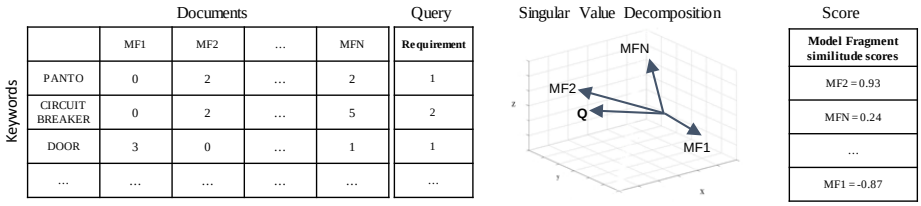


Fig. 4. Traceability Link Recovery through Latent Semantic Indexing Example

3 Evaluation

Through the following paragraphs, we present the research questions that our work tackles, describe our real-world case study and the oracle used for our experiment, detail the design of our experiment, and present the obtained results.

3.1 Research Questions

From the described problem, two research questions arise:

RQ1: How does the usage of different M2NL techniques affect the effectiveness and efficiency of TLR over requirements and models?

RQ2: How does the inclusion of either Simple or Advanced NLP techniques along with M2NL affect the effectiveness and efficiency of TLR over requirements and models?

3.2 Case study

For our experiment, CAF provided us with requirements and models of five railway solutions from Auckland, Bucharest, Cincinnati, Houston, and Kaohsiung. The trains are specified by about 100 requirements each, with an average of 50 words. Regarding models, trains are specified through an average 8250 model elements. CAF also provided lists of domain terms and stopwords. The domain terms list comprehends around 300 domain terms, and the stopwords list comprehends around 60 words. Both lists were created by a CAF domain expert associated to the provided products.

3.3 Oracle

In order to evaluate the results of our experiment, CAF provided us with their existing documentation on requirements-models traceability. Each requirement can be mapped to a single model fragment. A model fragment is a model elements subset, specified with the model fragment formalization capacities of the Common Variability Language (CVL) [7]. We use the existing traceability as the oracle for evaluating the impact of each of the M2LN-NLP configurations on LSI. To achieve this, we analyze the results of the rankings generated by LSI, checking the position of the ranking in which the oracle (correct model fragment for the input requirement) appears.

3.4 Design of the experiment

The first step is to select a M2NL-NLP configuration. With the chosen configuration, we extract the NL representation of the model fragments in our case study. Then, we perform the necessary NLP over the text of both all the requirements and all the NL representations of model fragments in our case study.

From the strings achieved through the first step, all the individual words are extracted to form a list of words. The list of words (*terms*), the processed representations of model fragments (*documents*), and one processed requirement (*query*), are used as input for LSI. LSI returns a ranking of model fragments, ordered according to their similarity to the requirement. LSI is performed several times, taking each requirement from our case study as *query*, in order to extract the model fragment rankings for all the available requirements. Through these rankings and the oracle, we can determine the ranking positions in which the correct model fragments appear for each requirement. Through the results, we are able to evaluate the impact of the chosen M2NL-NLP configuration over LSI.

The described steps (choosing a configuration, performing NLP of requirements and model fragments, LSI, impact analysis) are carried out four times, until the four configurations are chosen and analyzed.

3.5 Results

For each M2NL-NLP configuration, we measured the average, best, and worst result in the rankings generated by LSI. We also measured the time that the execution of M2NL-NLP took for the different configurations on average after 25 executions. We do not highlight the LSI execution time averages, since it is practically identical for all the configurations (around 70 seconds). Table 1 shows the results achieved by LSI when performed over the four configurations, with the best results highlighted in light gray.

Table 1. Results per M2NL-NLP techniques configuration

	Average Result ± Standard Deviation	Best Result	Worst Result	Time Taken (s)
Configuration 1	#2 ± 5.54	#1	#34	12
Configuration 2	#2 ± 5.09	#1	#30	296
Configuration 3	#1 ± 2.75	#1	#19	53
Configuration 4	#1 ± 1.12	#1	#5	336

Configuration 4 (Rule-Based M2NL + Advanced NLP) leads LSI to the best results, retrieving an average ranking position of #1 ± 1.12, a ranking position #1 as its best result, and a ranking position #5 as its worst. Configuration 1 (Element-Based M2NL + Simple NLP), on the opposite, is the one that leads LSI to the worst results. Its best result is position #1, but its worst peaks at position #34, presenting an average of #2 ± 5.54. Configurations 2 (Element-Based M2NL + Advanced NLP) and 3 (Rule-Based M2NL + Simple NLP) present intermediate values, being Configuration 3 slightly better than Configuration 2.

4 Statistical analysis

To properly compare the different configurations, the data resulting from the empirical analysis was analyzed using statistical methods.

4.1 Statistical significance

A statistical test must be run to assess whether there is enough empirical evidence to claim that there is a difference between two configurations (e.g., A is better than B). To achieve this, two hypotheses are defined: the null hypothesis H_0, and the alternative hypothesis H_1. The null hypothesis H_0 is typically defined to state that there is no difference among the configurations, whereas the alternative hypothesis H_1 states that the configurations differ. In such a case, a statistical test aims to verify whether the null hypothesis H_0 should be rejected.

The statistical tests provide a probability value, $p - value$. The $p - value$ obtains values between 0 and 1. The lower the $p - value$ of a test, the more likely that the null hypothesis is false. It is accepted by the research community that a $p - value$ under 0.05 is statistically significant [10], and so the hypothesis H_0 can be considered false. The carried test depends on the properties of the data. Since our data does not follow a normal distribution in general, our analysis requires the use of non-parametric techniques. There are several tests for analyzing this kind of data; however, the Quade test is the most powerful when working with real data [11]. In addition, according to Conover [12], the Quade test is the one that has shown the best results for a low number of configurations.

The $p - Value$ of this test is $4.208x10^{-6}$ and the statistic of this test is 9.659. Since the $p - Value$ is smaller than 0.05, we reject the null hypothesis. Consequently, we can state that there exist differences between among the four configurations for the performance indicator of the position in the ranking.

However, with the Quade test, we cannot answer the following question: *Which of the configurations gives the best performance?* In this case, the performance of each configuration should be individually compared against all other alternatives. In order to do this, we perform an additional post hoc analysis. This kind of analysis performs a pair-wise comparison among the results of each configuration, determining whether statistically significant differences exist among the results of a specific pair of configurations.

The second column of Table 2 shows the $p - Values$ of Holm's post hoc analysis for the performance indicator and the specific pair of configurations (e.g., Configuration 1 and Configuration 2). The $p - Values$ shown in this table for two comparisons (the Configuration 1 vs Configuration 2 and Configuration 3 vs Configuration 4) are greater than the corresponding significance threshold value (0.05), whereas the $p - Values$ for the other comparisons are smaller than 0.05. Hence, we can determine that the differences in performance between the Configuration 1 vs Configuration 2 and the Configuration 3 vs Configuration 4 are not significant, but the differences in performance are significant in the other comparisons (e.g., the comparison shows significant differences between the Configuration 2 vs Configuration 4).

4.2 Effect size

Statistically significant differences can be obtained even if they are so small as to be of no practical value [10]. Therefore, it is important to assess if a configuration is statistically better than another and to assess the magnitude of the improvement. *Effect size* measures are taken in account to analyze this phenomenon.

For a non-parametric effect size measure, we use Vargha and Delaney's $\hat{A}_{12}$ [13]. $\hat{A}_{12}$ measures the probability that running one configuration yields higher values than running another configuration. If the two configurations are equivalent, then $\hat{A}_{12}$ will be 0.5.

For example, $\hat{A}_{12} = 0.7$ means that we would obtain better results in 70% of the runs with the first of the pair of configurations that have been compared, and $\hat{A}_{12} = 0.3$ means that we would obtain better results in 70% of the runs with the second of the pair of configurations that have been compared. Thus, we have an $\hat{A}_{12}$ value for every pair of configurations.

Table 2. Holm's post hoc $p - Values$ and $\hat{A}_{12}$ statistic for each pair of configurations

Pair of configurations	$p - Value$	$\hat{A}_{12}$
Configuration 1 vs Configuration 2	0.89	0.4983
Configuration 1 vs Configuration 3	0.0065	0.3948
Configuration 1 vs Configuration 4	0.00079	0.4005
Configuration 2 vs Configuration 3	0.0024	0.3883
Configuration 2 vs Configuration 4	$3.4x10^{-5}$	0.3973
Configuration 3 vs Configuration 4	0.83	0.5122

The third column of Table 2 shows the values of the effect size statistics between every pair of configurations.

The $\hat{A}_{12}$ values show a slight superiority (even though these values are closer to the equivalent value of 0.5) of the Configuration 2 in the comparison with the Configuration 1, and the Configuration 3 in the comparison with the Configuration 4. The $\hat{A}_{12}$ values show the largest differences, with values around 0.39 when Configuration 1 and Configuration 2 are compared with Configuration 3 or Configuration 4.

Overall, these results confirm that Configuration 3 and Configuration 4 outperform Configuration 1 and Configuration 2.

5 Threats to Validity

In this section, we use the classification of threats of validity of [14] 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. In order to minimize this risk, we study the positions of the oracles in the rankings, an objective and widely accepted measure, used before by other researchers in the community [15].
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 finding, and to what extent the findings are of relevance for other cases. Both requirements and conceptual 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. Therefore, our experiment does not rely on the particular conditions of our domain. Nonetheless, 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. The requirements and models of the trains used through our experiment were provided by our industrial partner engineers, as well as the domain terms and stopwords lists, which were crafted by domain experts not involved in this research.

6 Related Work

In [16], NLP techniques are used to assess equivalence between requirements. The authors conclude that performance of NLP in their field is determined by the properties of the provided datasets. Properties are then considered as a factor to adjust the NLP process and performance over an industrial case study. Through our work, rather than adjusting the NLP process to study equivalence between requirements, we tackle the impact of different M2NL-NLP configurations on LSI, exposing the way they behave and improve (or worsen) the IR process.

The work presented in [17] uses NLP to study how changes in requirements impact other requirements. The authors analyze TLR between requirements, and use NLP to determine the propagation of changes. Our work does not analyze changes in requirements or how they affect the system, but rather on what is the most appropriate way of applying M2NL-NLP to requirements-models TLR. Moreover, the authors of [17] do not consider different NLP configurations, but rather guide the process through requirements properties.

Finally, [18] takes in consideration the possible LSI configurations for TLR between requirements and test cases. The authors state that LSI configurations depend on the available datasets, and also look forward to automatically determining appropriate LSI configurations for any given dataset. We do not tackle the impact of using different LSI configurations for TLR, but rather analyze how different M2NL-NLP configurations affect the results of TLR.

7 Concluding Remarks

Through this paper, we analyze how different M2NL-NLP techniques impact the outcome of requirements-models TLR. To that extent, we process the requirements and models that specify a real-world industrial case study through a series of combinations of M2NL-NLP techniques, and then perform Latent Semantic Indexing (LSI) over the processed specifications. We study the rankings produced by LSI with our oracle to evaluate the impact of the M2NL-NLP techniques over TLR. Results show that:

1 Rule-Based M2NL improves the results of Element-Based M2NL in a statistically significant manner, but it requires an additional effort from software engineers when Domain Specific Languages (DSLs) are used. The rules from [1] are specific for UML models, and work with the TCML DSL of our industrial partner due to it being derived from UML, but engineers that use a non-UML DSL need to either adapt the existing rules or create DSL-specific rules. With the obtained results, engineers have more information to choose between investing their efforts in Rule-Based M2NL, or (in case it yields sufficiently reliable TLR results) using Element-Based M2NL.

2 The usage of Advanced NLP along with M2NL always improves its results, although in a non-statistically significant manner. We noticed that the terms used in the conceptual models are close to those of requirements, so Advanced NLP does not have a huge impact over the results. Nonetheless, the application of the Advanced NLP techniques does not require a huge effort, and therefore its application can be deemed worthy when maximizing the quality of TLR results is a key priority.

Acknowledgements

This work has been partially supported by the Ministry of Economy and Competitiveness (MINECO) through the Spanish National R+D+i Plan and ERDF funds under the project Model-Driven Variability Extraction for Software Product Line Adoption (TIN2015-64397-R). We also thank ITEA3 15010 REVaMP2 Project.

References

1. Meziane, F., Athanasakis, N., Ananiadou, S.: Generating Natural Language Specifications from UML Class Diagrams. Requirements Engineering **13**(1)
2. Font, J., Arcega, L., Haugen, Ø., Cetina, C.: Feature Location in Models through a Genetic Algorithm driven by Information Retrieval Techniques. In: Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems
3. Hulth, A.: Improved Automatic Keyword Extraction given more Linguistic Knowledge. In: Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing
4. Plisson, J., Lavrac, N., Mladenic, D., et al.: A Rule Based Approach to Word Lemmatization. In: Proceedings of the 7th International Multi-Conference Information Society. Volume 1.
5. Landauer, T.K., Foltz, P.W., Laham, D.: An Introduction to Latent Semantic Analysis. Discourse processes **25**(2-3)
6. Poshyvanyk, D., Gueheneuc, Y.G., Marcus, A., Antoniol, G., Rajlich, V.: Feature Location Using Probabilistic Ranking of Methods Based on Execution Scenarios and Information Retrieval. IEEE Transactions on Software Engineering **33**(6)
7. Haugen, Ø., Møller-Pedersen, B., Oldevik, J., Olsen, G.K., Svendsen, A.: Adding Standardized Variability to Domain Specific Languages. In: 12th International Software Product Line Conference
8. Zhang, X., Haugen, O., Moller-Pedersen, B.: Model Comparison to Synthesize a Model-Driven Software Product Line. In: 15th International Software Product Line Conference
9. 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
10. Arcuri, A., Briand, L.: A Hitchhiker's Guide to Statistical Tests for Assessing Randomized Algorithms in Software Engineering. Softw. Test. Verif. Reliab. **24**(3)
11. García, S., Fernández, A., Luengo, J., Herrera, F.: Advanced Nonparametric Tests for Multiple Comparisons in the Design of Experiments in Computational Intelligence and Data Mining: Experimental Analysis of Power. Inf. Sci. **180**(10)
12. Conover, W.J.: Practical Nonparametric Statistics, 3rd Edition. Wiley (1999)
13. Vargha, A., Delaney, H.D.: A Critique and Improvement of the CL Common Language Effect Size Statistics of McGraw and Wong. Journal of Educational and Behavioral Statistics **25**(2)
14. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering. Springer Science & Business Media (2012)
15. Haiduc, S., Bavota, G., Marcus, A., Oliveto, R., De Lucia, A., Menzies, T.: Automatic Query Reformulations for Text Retrieval in Software Rngineering. In: 35th International Conference on Software Engineering
16. Falessi, D., Cantone, G., Canfora, G.: Empirical Principles and an Industrial Case Study in Retrieving Equivalent Requirements via Natural Language Processing Techniques. IEEE Transactions on Software Engineering **39**(1)
17. Arora, C., Sabetzadeh, M., Goknil, A., Briand, L.C., Zimmer, F.: Change Impact Analysis for Natural Language Requirements: An NLP Approach. In: 23rd International Requirements Engineering Conference
18. Eder, S., Femmer, H., Hauptmann, B., Junker, M.: Configuring Latent Semantic Indexing for Requirements Tracing. In: Proceedings of the Second International Workshop on Requirements Engineering and Testing