Taxonomic Trace Links Recommender: Context Aware Hierarchical Classification

Waleed Abdeen[1,*,†]

[1] *Blekinge Institute of Technology, Karlskrona, Sweden.*

Abstract

In the taxonomic trace links concept, the source and target artifacts are connected through knowledge organization structure (e.g., taxonomy). We introduce in this paper a recommender system that recommends labels to requirements artifacts from domain-specific taxonomy to establish taxonomic trace links. The tool exploits the hierarchical nature of taxonomies and uses requirements text and context information as input to the recommender.

Keywords

Requirements traceability, Taxonomy, Hierarchical classification, Recommender system

1. Introduction

Requirements traceability is the capability of tracing the requirement life in a backward and forward manner [1]. The adoption of requirements traceability has proven to lead to lesser defects in the produced software [2]. Traditional trace links connect the source and target artifacts using a direct link. An alternative approach is the taxonomic trace links, in which the source and target artifacts are traced through a knowledge organization structure (e.g., taxonomy or ontology) [3]. The taxonomic trace link approach has three main advantages over traditional techniques: tracing artifacts with different abstractions, better structure of artifacts used in the development process, and early establishment of trace links.

Unterkalmsteiner [3] introduced the concept of taxonomic trace links in a trace links recommender system. The initial evaluation [4] showed that the tool had low confidence in the recommended labels. In this paper, we present a tool implementation of the taxonomic trace links with adaptation by using hierarchical text classification and adding the context information of a requirement to the tool inputs.

In: A. Ferrari, B. Penzenstadler, I. Hadar, S. Oyedeji, S. Abualhaija, A. Vogelsang, G. Deshpande, A. Rachmann, J. Gulden, A. Wohlgemuth, A. Hess, S. Fricker, R. Guizzardi, J. Horkoff, A. Perini, A. Susi, O. Karras, A. Moreira, F. Dalpiaz, P. Spoletini, D. Amyot. Joint Proceedings of REFSQ-2023 Workshops, Doctoral Symposium, Posters & Tools Track, and Journal Early Feedback Track. Co-located with REFSQ 2023. Barcelona, Catalunya, Spain, April 17, 2023.

*Corresponding author.

waleed.abdeen@bth.com (W. Abdeen)

https://waleedabdeen.com (W. Abdeen)

0000-0001-8142-9631 (W. Abdeen)

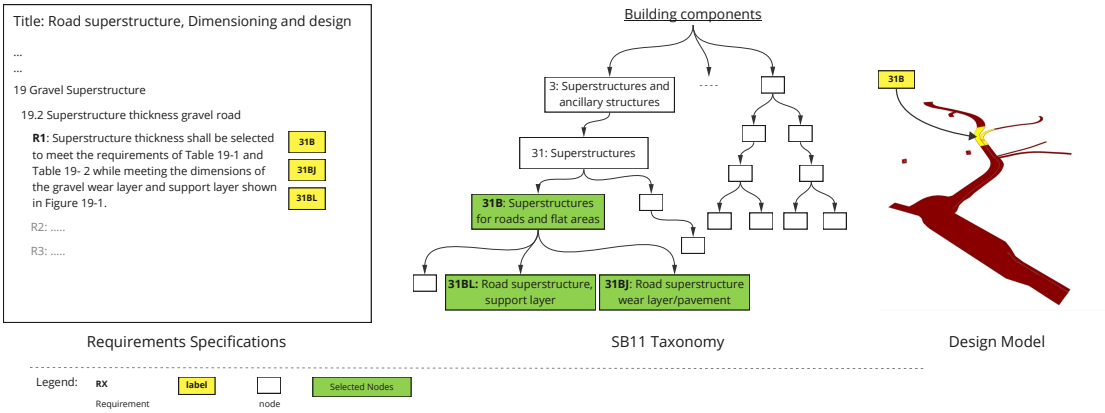

Figure 1: Taxonomic Trace Links Example

2. Background

We explain in this section the taxonomic trace link and the hierarchical classification concepts, which are important for the reader to understand the tool process.

Taxonomic trace links concept was introduced earlier by Unterkalmsteiner [3]. The main idea is to label the traced source and target artifacts with nodes from a domain-specific taxonomy in order to trace these artifacts. In the same study, the author presented a prototype that materializes the taxonomic trace links concept. The prototype finds relevant labels for a requirement from a taxonomy. The early evaluation of the tool showed that the approach of taxonomic trace links is attractive for practitioners. However, better recommendations are required [4].

Figure 1 illustrates an example of taxonomic trace links in the context of the construction domain. Assuming that we want to create trace links from the requirements specifications to the design model using SB11 taxonomy, which is used to classify objects in the construction domain and contains 2074 nodes. The yellow boxes (31B, 31BJ, 31BL) are labels that we set on the requirement **R1**. These labels are nodes from the taxonomy, highlighted in green. Similarly, an object of the design model is labeled with 31B. In conclusion, a trace link is created between the requirement **R1** and the highlighted object in the design model because both are labeled with the same node (31B).

Hierarchical text classification is the use of hierarchical classification space to classify a document using one or more classes [5]. The topic was studied by many researchers and showed to improve the performance of the classification [5, 6, 7, 8, 9, 10]. Song et al. [9] have proposed an approach of dateless hierarchical text classification, where the training of the model does not require any labeled data. They classify newsgroups messages and newswire stories using a taxonomy with a depth level of two. Their problem is similar to ours, where they use taxonomy to classify text with no training data. Furthermore, the evaluation of the algorithm showed that its performance is comparable with those of the supervised learning algorithms [9]. Thus,

we implement the algorithm presented by the authors in our tool, with some adaptations, to classify requirements text. We explain the adaptations in Section 4.

The novelty of our tool over the previous implementation [3] lies in:

1. The use of hierarchical text classification: we argue that using hierarchical text classification in the taxonomic trace link implementation could yield better results. In the previous studies of taxonomic trace links [3, 4] a flat classifier was adopted where all nodes in the taxonomy are treated as if they were on one level. On the contrary, we use a hierarchical classifier that considers the hierarchy of the taxonomy.
2. The use of context information: when labeling requirements using a domain-specific taxonomy, the participants used context information (document and section titles) to choose the true label from the taxonomy [4], the previous classifier did not use this information as input. Furthermore, the use of context information is proven to improve the performance of text classification methods [11].

3. Tool Development Context

This tool was developed as a part of the project D-CAT in collaboration with Trafikverket, Swedish Transport Authority. In construction projects, a 3D presentation of the physical system is first produced before building the actual system. It is important that the produced design models adhere to the specified requirements, known as rules and regulations, as it is much cheaper to redraw a design model than rebuild a bridge. The verification of these requirements in the design models becomes tedious when traceability is missing between requirements and design models. At the time of writing this paper, traceability was not practiced in Trafikverket due to the lack of digitization.

We aim in this tool to implement traceability that is usable in a similar context, where the software or system must adhere to a list of requirements. During the tool development, we used natural language requirements from the construction domain and the SB11 taxonomy. The tool implementation makes it possible to replace the taxonomy with one from a different domain.

4. The Taxonomic Trace Links Recommender System

The source code of the tool and the required data are available online in the replication package [1]. The tool recommends K labels from a domain-specific taxonomy to requirements artifacts. We present a screenshot of the tool's user interface in Figure 2. The following steps need to be followed when using the tool:

1. The user inputs the **requirement text** and any relevant context information, currently the user can input the **document title** of the software requirements specification (SRS) and **section title(s)** under which the requirement is listed. Then, the user clicks on classify.

[1] https://zenodo.org/record/7638518

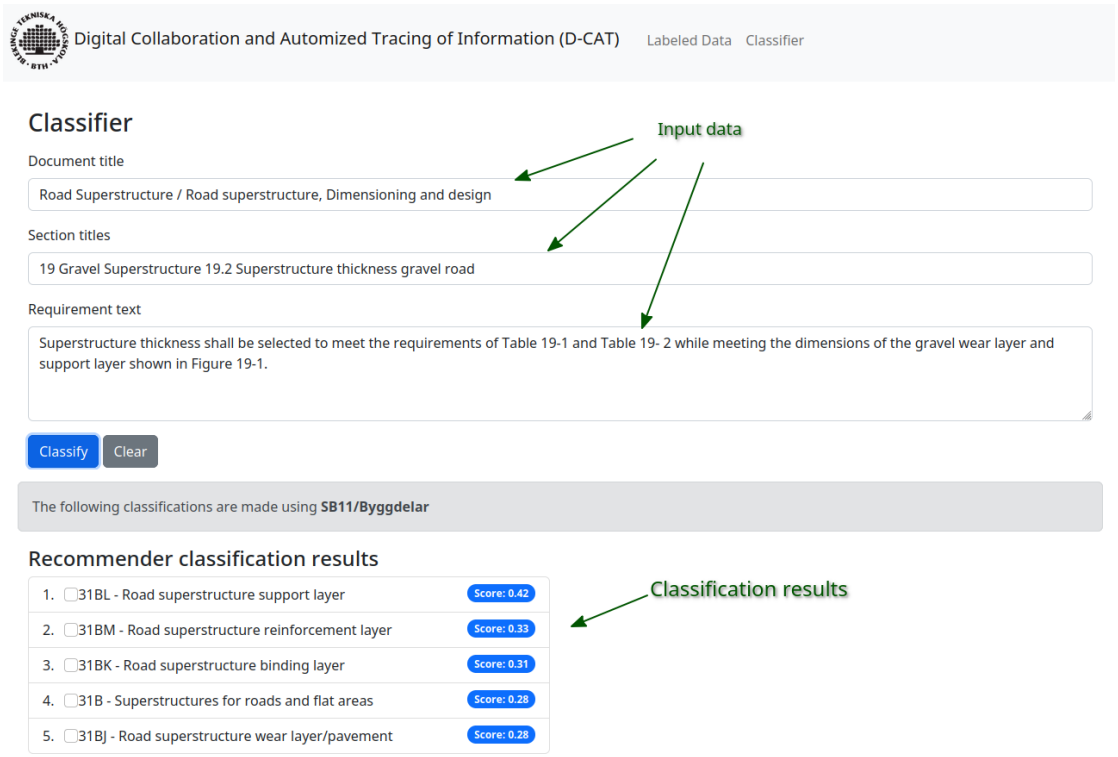

Figure 2: Recommender Interface

2. The tool processes the input from the user and identifies the most relevant labels based on the text similarity between the requirement and the labels of the taxonomy (we explain how the similarity is calculated later in this section).
3. The recommended labels are retrieved and shown, in a descending order based on the similarity score, under the section **recommender classification results**. A recommended label is displayed using the label code, label description, and a similarity score.A recommended label could be either correct or incorrect.
4. The user makes the final decision by choosing the best recommendations as they see fit from the list of results.

Figure 3, depicts the internal recommender process, which consists of three main steps: 1) taxonomy conceptualization, 2) requirement conceptualization, and 3) requirement classification. Each main step consists of several sub-steps represented with yellow-filled rectangles.

4.1. Taxonomy Conceptualization

In this step, the domain-specific taxonomy nodes are **preprocessed**, by removing the stop-words from each node description. Then, **labels are aggregated** to achieve a hierarchical classification

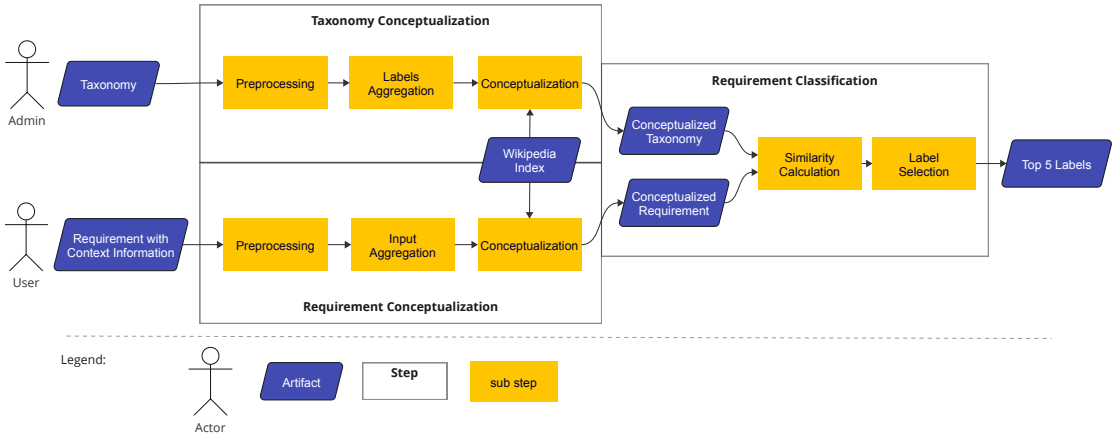

Figure 3: The Recommender Process

using the top-down approach, the description of each node in the taxonomy is brought up to the parent node. Consequently, a node description contains its own node description and the description of all its child nodes. Finally, **conceptualization** is performed on the taxonomy. A concept vector is created for every node using explicit semantic analysis (ESA) [9], which uses indexed Wikipedia articles as an external knowledge base to find relevant concepts for a given text by calculating TFIDF score for words. The output is conceptualized taxonomy nodes, each presented using a vector of concepts.

4.2. Requirements Conceptualization

The user enters a requirement text and the relevant context information (document and section titles). First, the tool **preprocesses** the data by removing stop words. Then the requirement text is aggregated with the context information (**input aggregation**). Finally, a **conceptualization** of the aggregated input is performed, similar to the conceptualization sub-step in taxonomy conceptualization in Section 4.1. The output is a conceptualized requirement presented as a vector of concepts.

4.3. Requirements Classification

The outputs of the two previous steps (conceptualized taxonomy and conceptualized requirement) are taken as inputs to the requirement classification step. A **similarity calculation** is performed between both inputs as follows:

$$score = cos(\phi_x(l_i), \phi_x(r)) \tag{1}$$

where ϕ_x is the ESA representation of a text, r is the aggregated requirement text, and (l_i) is node i in the taxonomy. The similarity score is normalized in the range 0-1. Finally, in **label selection**, we choose not to report the top labels at each level as done by Song et al. [9], but rather we chose to select the top labels from all nodes. The reason behind this decision is that we

are interested in recommending the most relevant label at the deepest level, which represents the most concrete label. The output of the tool is the top **K** recommended labels to classify the requirement entered by the user. We refer the reader to Song et al. [9] paper for more details about the algorithm used for classification.

5. Planned Evaluation

We plan to conduct lab experiments to evaluate the recommender system performance (presented in this paper) and compare it with the recommender system that was developed by Unterkalmsteiner [3]. The planned experiments will be conducted by manipulating two independent variables, namely the taxonomy and the classifier (hierarchical and flat). We chose the taxonomy as an independent variable to understand: if and to what the use of different taxonomy could affect the performance of the recommender system. The dependent variable is the recommender performance and will be presented by the set of metrics, precision, recall, and F1 score. A dataset of natural language requirements that are labeled with nodes from domain-specific taxonomy is needed to measure the performance of the recommender. We are currently working, with domain experts, on developing a ground truth of 129 requirements sampled from the construction domain.

After conducting the lab experiments and fine-tuning the tool, the next step is to pilot the tool in a real project with practitioners to evaluate it in natural settings.

6. Conclusion

We have presented the recommender system, which uses a domain-specific taxonomy to recommend labels to requirements artifacts. This classification aims to establish taxonomic trace links between requirements artifacts and other software artifacts. This is the first version of the tool, and we are working on improving the tool in an iterative manner using a design science framework. We have developed and demonstrated the tool in the context of construction projects. The implementation of the tool makes it possible to use in software projects as 1) the recommender classifies natural language requirements, and 2) the taxonomy can be replaced with one from a different domain during the setup of the tool. However, the tool needs to be evaluated in controlled and natural settings in the context of software projects and adapted based on the outcome of the evaluation.

The planned future work is to 1) improve the tool in terms of calculating the similarity score between a requirement artifact and the nodes from the domain-specific taxonomy, 2) integrate the tool with existing requirements management tools, e.g., DOORS, 3) evaluate the performance of the recommender system on a set of software requirements and further evaluate the retrieval of trace links.

Acknowledgement

This research was funded by DCAT project. Thanks to Dr. Michael Unterkalmsteiner and Dr. Krzysztof Wnuk for their support and feedback.

References

[1] O. Gotel, C. Finkelstein, An analysis of the requirements traceability problem, in: Proceedings of IEEE International Conference on Requirements Engineering, 1994, pp. 94–101. doi:10.1109/ICRE.1994.292398.

[2] P. Rempel, P. Mäder, Preventing defects: The impact of requirements traceability completeness on software quality, IEEE Transactions on Software Engineering 43 (2017) 777–797. doi:10.1109/TSE.2016.2622264, conference Name: IEEE Transactions on Software Engineering.

[3] M. Unterkalmsteiner, TT-RecS: The taxonomic trace recommender system, in: 2020 IEEE Seventh International Workshop on Artificial Intelligence for Requirements Engineering (AIRE), 2020, pp. 18–21. doi:10.1109/AIRE51212.2020.00009.

[4] M. Unterkalmsteiner, Early requirements traceability with domain-specific taxonomies - a pilot experiment, in: 2020 IEEE 28th International Requirements Engineering Conference (RE), 2020, pp. 322–327. doi:10.1109/RE48521.2020.00042, ISSN: 2332-6441.

[5] A. Sun, E.-P. Lim, Hierarchical text classification and evaluation, in: Proceedings 2001 IEEE International Conference on Data Mining, 2001, pp. 521–528. doi:10.1109/ICDM.2001.989560.

[6] L. Cai, T. Hofmann, Hierarchical document categorization with support vector machines, in: Proceedings of the thirteenth ACM international conference on Information and knowledge management, CIKM '04, Association for Computing Machinery, 2004, pp. 78–87. URL: https://doi.org/10.1145/1031171.1031186. doi:10.1145/1031171.1031186.

[7] S. Gopal, Y. Yang, Recursive regularization for large-scale classification with hierarchical and graphical dependencies, in: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '13, Association for Computing Machinery, 2013, pp. 257–265. URL: https://doi.org/10.1145/2487575.2487644. doi:10.1145/2487575.2487644.

[8] L. Xiao, D. Zhou, M. Wu, Hierarchical classification via orthogonal transfer, in: L. Getoor, T. Scheffer (Eds.), Proceedings of the 28th International Conference on Machine Learning (ICML-11), ICML '11, ACM, 2011, pp. 801–808. Event-place: Bellevue, Washington, USA.

[9] Y. Song, D. Roth, On dataless hierarchical text classification, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 28, 2014.

[10] H. Peng, J. Li, Y. He, Y. Liu, M. Bao, L. Wang, Y. Song, Q. Yang, Large-scale hierarchical text classification with recursively regularized deep graph-CNN, in: Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18, ACM Press, 2018, pp. 1063–1072. URL: http://dl.acm.org/citation.cfm?doid=3178876.3186005. doi:10.1145/3178876.3186005.

[11] Y. Wang, C. Wang, J. Zhan, W. Ma, Y. Jiang, Text FCG: Fusing contextual information via graph learning for text classification, Expert Systems with Applications (2023) 119658. URL: https://linkinghub.elsevier.com/retrieve/pii/S0957417423001598. doi:10.1016/j.eswa.2023.119658.