*Nuclear Science and Technology Symposium - SYP2025*
*Espoo, Finland, 21-22 October 2025*

# Utilizing Large Language Models to Improve Requirements Tracing in the Nuclear Energy Domain

**Pyry Aho**
Fortum Power and Heat Oy, Nuclear Generation
Keilalahdentie 2-4, 02150 Espoo, Finland
pyry.aho@fortum.com

## ABSTRACT

This paper summarizes the results of a thesis conducted in collaboration with Fortum's Nuclear Generation business unit and Aalto University. The aim of the thesis was to investigate how large language models can be utilized to improve the requirements tracing process in the nuclear energy domain. To achieve this, the design science method was applied to create a prototype for automated candidate trace link generation. The prototype used a similarity-based approach for generating candidate trace links, where LLMs were utilized to compute the similarity between two requirements (or a requirement and a document section). Based on the interviews with four experts at Fortum's Nuclear Generation business unit, two use cases were defined for the prototype: requirement-to-requirement tracing and requirement-to-documentation tracing. The prototype's accuracy in the first use case was on a decent level, finding around 80% of the true links in its top-5 candidate links in three out of the four test sets. However, the limitations of the prototype were highlighted in the requirement-to-documentation use case, where the prototype was only able to find around 50% of the true links in its top-5 candidate links.

## 1 INTRODUCTION

Nuclear power plants are complex safety-critical systems and projects related to them are highly regulated. Therefore, modernization and renewal projects have a large number of requirements derived from regulations and guidelines. With the addition of the technical requirements imposed by the existing systems and devices in the plant, large projects can have thousands of requirements, which need to be traced throughout the project life cycle.

Requirements traceability is defined as "the ability to describe and follow the life of a requirement, in both a forwards and backwards direction" [1, p. 4]. This definition implies that requirements should be able to be traced both backwards to their origins and forwards to all documentation created based on the requirements [1]. The requirements tracing process can be divided into four steps: document parsing, candidate link generation, candidate link evaluation and traceability analysis [2]. According to Sultanov et al. [2], candidate link generation refers to the process of finding "candidate" links between the requirements and other design artifacts. Depending on the approach, candidate link generation can be a repetitive, mundane and time-consuming activity, especially if the amount of requirements is large [3].

The candidate link generation process has previously been automated via the use of information retrieval methods [3]. However, commonly used information retrieval (IR) methods, such as the vector space model (VSM) and latent semantic indexing (LSI), often struggle with synonyms or abbreviations [4]. Furthermore, IR methods often use the bag-of-words representation which ignores the order and context of the terms in the sequence [5].

Statistical language models enable the utilization of the ordering and context between terms. These models use n-gram approximations and the context of the previous $n$ terms in the sequence to calculate the probability that a query is relevant to the document [6]. However, as DeLucia et al. [6] point out, the computational cost of estimating these probabilities grows exponentially as the context size, $n$, is increased.

Using large language models (LLMs) to determine the similarity between two pieces of text offers a promising solution for overcoming the challenges with IR methods and statistical language models. In contrast to statistical language models, LLMs are able to utilize much larger context windows while being easily parallelizable (i.e., the computations can be processed concurrently rather than sequentially) [7]. Additionally, LLMs are trained using massive amounts of text, which allows the model to generalize to unseen data and to learn similarities between different terms [7].

This paper presents the development and evaluation of a prototype for automated requirements tracing for generating requirement-to-requirement and requirement-to-documentation candidate trace links. Different methods for automated requirements tracing were compared, focusing on VSM and similarity-based LLM methods.

## 2 RESEARCH METHODS

The main contributions of the thesis were a literature review and an empirical study. *Design science* method was applied in the empirical study to design and implement a first prototype for automated candidate trace link generation.

### 2.1 Research problem

The aim of the thesis was to present and implement a semi-automated method to assist in the current requirements tracing process. Large language models were identified as a promising technical solution for automating parts of the process. Therefore, the research problem was formulated as: *How can large language models be utilized to improve the requirements tracing process in the nuclear energy domain?*

### 2.2 Literature review

A systematic literature review was conducted to collect and analyse scientific literature relevant to the research problem of the thesis. The scope of the literature review was limited to requirements tracing, requirements management in the nuclear energy domain, and the utilization of large language models in text similarity analysis.

The search and selection of relevant literature was done systematically. The search and selection process consisted of three main steps: trial-and-error search to find suitable search queries, database search based on the queries, and applying the snowballing method to the papers found from the database search.

### 2.3 Design science

The empirical study applied the design science research method, which "creates and evaluates IT artifacts intended to solve identified organizational problems" [8, p. 49]. In the context of the thesis, the organizational problem was the time-consuming manual requirements tracing process. The proposed IT artifact to solve this problem was the prototype for automated candidate link generation.

The design science process consisted of five main steps introduced by Peffers et al. [8]. *The problem identification step* of the empirical study consisted of conducting semi-structured interviews and a document analysis. In the *definition of the solution objectives step*, the results of the interviews and document analysis were used to determine the prototype's main objectives and detailed requirements.

*The design and development step* consisted of designing the architecture of the prototype based on functional and architecturally significant non-functional requirements. In *the demonstration and evaluation step*, the prototype was evaluated using both quantitative and qualitative measures. The quantitative evaluation included testing the prototype's accuracy and performance on multiple different test sets. The qualitative evaluation included evaluating the fulfilment of the prototype's requirements.

*The communication step* included presenting the prototype and the results of its evaluation to the main stakeholders of the AI-project group at Fortum and the thesis seminar arranged by the university.

### 2.4 The prototype

The prototype was trained using the sentence-transformers library and its API for fine-tuning LLMs, and existing requirements tracing data from earlier projects. Two versions of the RoBERTa LLM fine-tuned for general text similarity tasks were tested. The smaller *all-distilroberta-v1* model had 82.1 million parameters and the larger *all-roberta-large-v1* 355 million parameters. These relatively small models were used, because the prototype had to be run locally due to the confidentiality of the training and testing data.

Figure 1 shows the fine-tuning pipeline of the LLM. The model takes two sections of text (a requirement or document section) as its input and predicts, whether a trace link should exist between the two inputs. These predictions are compared against the manually created trace links in the training data. Then, the parameters of the model are tuned to improve the predictions, i.e. give higher similarity scores to inputs, which have a trace link between them.

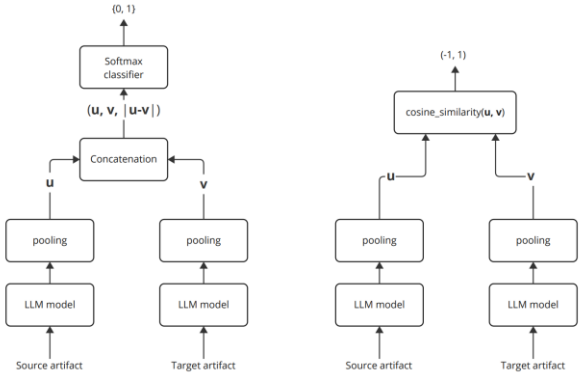


Figure 1: The fine-tuning (left) and inference (right) pipelines LLM, modified from [9].

Recall was used as the main accuracy metric for the evaluation of the prototype. Recall is defined

as the ratio between found true links and all true links. The recall value approaches one as the amount candidate links is increased (if all possible candidate links are returned, the recall value is always 1). Therefore, recall at different intervals of returned candidate links was used to evaluate the prototype.

# 3 RESULTS

## 3.1 Literature Review

The most relevant findings of the literature review in terms of the development of the prototype and their effects on the empirical study are gathered in Table 1. These discoveries were used in the "*defining the solution objectives*" phase of the empirical study, when defining the high-level objectives and requirements for the prototype. Additionally, the discoveries were utilized when designing the architecture of the prototype, choosing the training methods, and choosing the appropriate pre-trained LLM for the task.

Table 1: The main discoveries of literature review.

| Main discoveries | Effects on the empirical study |
|---|---|
| STUK requires that appropriate requirements traceability is implemented through all project phases. | Justifies the importance and need for implementing requirements tracing in projects. |
| Recall, precision and the F2-score are the main measures of accuracy used for evaluating automated requirements tracing methods. | Recall was used as the main accuracy metric for the evaluation of the prototype. |
| The requirements tracing problem can be formulated as a text matching problem. | The prototype's similarity-based LLM implementations followed the text matching problem formulation. |
| Fine-tuning LLMs on domain-specific data can considerably increase the accuracy of the results. | Requirements tracing data from earlier projects was used to fine-tune the pre-trained LLMs. |
| Autoencoder models generally outperform autoregressive models in natural language understanding tasks. | Autoencoder models (RoBERTa) were used in the implementation of the prototype. |

## 3.2 Empirical Study

The empirical study tested a similarity-based approach for generating requirement-to-requirement and requirement-to-documentation candidate trace links. The basic idea behind the approach was to utilize LLMs to calculate the similarity between two inputs (requirement or document section) and rank all the input pairs based on their similarity score. The process of calculating the similarity scores is modelled in the inference pipeline in Figure 1.

Three main implementations for computing the similarity were tested: VSM, basic LLM and fine-tuned LLM. Of these the fine-tuned LLM implementation performed the best on average. However, on the more challenging test sets (requirement-to-documentation tracing) all implementations seemed to perform equally poorly.

The prototype's accuracy was on a good level when creating trace links between native and process requirements (see Figure 2), and between technical stakeholder requirements and technical system-level requirements. In these cases, when retrieving the top-5 candidate links, the prototype averaged 80% recall. However, in one dataset where technical stakeholder requirements were linked to native requirements, the prototype struggled to find the correct links.

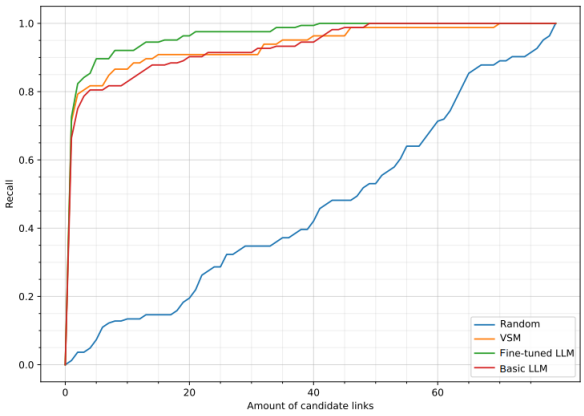


Figure 2: The recall of the prototype in requirement-to-requirement tracing.

As shown in Figure 3, the prototype's accuracy was significantly worse in the requirement-to-documentation use case than in the requirement-to-requirement use case. On average the model only found around 50% of the true links when the top-5 candidate links were considered.

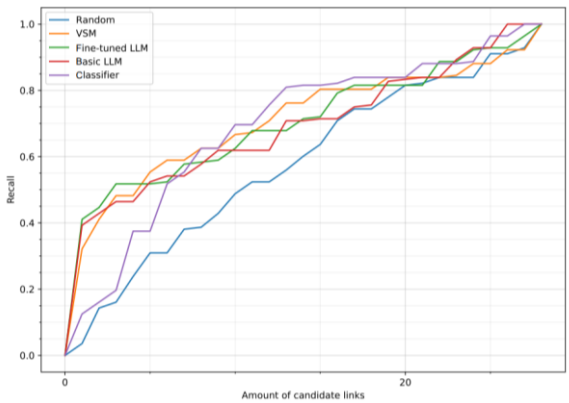


Figure 3: The recall of the prototype in requirement-to-documentation tracing.

## 4 CONCLUSIONS

Large language models seem to produce significantly more accurate candidate trace links when the source and target requirements are expressed in a similar level of technical detail. The prototype was able to generate sufficiently accurate trace links between process requirements and native requirements as well as between technical stakeholder requirements and technical system-level requirements. However, the prototype struggled to find accurate trace links between technical stakeholder requirements and native requirements (YVL requirements).

A similarity-based LLM approach may not be suitable for generating trace links between requirements and documentation. The results showed that the accuracy of the prototype was insufficient in requirement-to-documentation tracing. In theory, the vector representations of the requirements and document sections created by the LLM should be able to capture complex relationships between requirements and documentation. However, in practice the model struggled with the varying terminology and level of technical detail between the requirements and documentation.

As the thesis was limited to only testing relatively small autoencoder LLMs, future research could focus on testing the performance of state-of-the-art LLMs in the requirements tracing task. However, the recent development of LLMs has focused on generative AI and autoregressive models (such as GPT-4). Utilizing these models with the prototype would require considerable modifications.

A large part of the documentation and requirements produced at the case company are written in Finnish. Therefore, it could be beneficial to extend the tool's functionality to requirements and documentation written in Finnish. Furthermore, BERT models trained on Finnish datasets, such as FinBERT [10] from TurkuNLP, already exist and are freely available.

## ACKNOWLEDGEMENTS

The thesis was conducted in collaboration with Fortum's Nuclear Generation business unit and Aalto University. The author extends their gratitude to the thesis supervisor Prof. Fabian Fagerholm and advisors Dr. Marjo Kauppinen, Tapani Raunio (MSc) and Leena Kappinen (MSc), and the Nuclear AI project group for their invaluable feedback and help throughout the thesis process.

## REFERENCES

[1] O. Gotel and C. Finkelstein, "An analysis of the requirements traceability problem", Proceedings of IEEE International Conference on Requirements Engineering, 1994, pp. 94–101.

[2] H. Sultanov and J. H. Hayes, "Application of swarm techniques to requirements engineering: Requirements tracing", 18th IEEE International Requirements Engineering Conference, 2010, pp. 211–220.

[3] J. Hayes, A. Dekhtyar, and J. Osborne, "Improving requirements tracing via information retrieval", Proceedings. 11th IEEE International Requirements Engineering Conference, 2003., 2003, pp. 138–147.

[4] J. Cleland-Huang, A. Czauderna, M. Gibiec, and J. Emenecker, "A machine learning approach for tracing regulatory codes to product specific requirements", Proceedings of the 32nd ACM/ IEEE International Conference on Software Engineering - Volume 1, 2010, pp. 155–164.

[5] M. Borg, P. Runeson, and A. Ardö, "Recovering from a decade: A systematic mapping of information retrieval approaches to software traceability", Empirical Software Engineering, vol. 19, no. 6, pp. 1565–1616, 2014.

[6] A. De Lucia, A. Marcus, R. Oliveto, and D. Poshyvanyk, "Information retrieval methods for automated traceability recovery", Software and Systems Traceability, J. Cleland-Huang, O. Gotel, and A. Zisman, Eds. London: Springer London, 2012, pp. 71–98.

[7] D. Jurafsky and J. H. Martin, "Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition." unpublished, 2023.

[8] K. Peffers, T. Tuunanen, M. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research", Journal of Management Information Systems, vol. 24, pp. 45–77, Jan. 2007.

[9] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks", Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Nov. 2019.

[10] A. Virtanen, J. Kanerva, R. Ilo, et al., "Multilingual is not enough: Bert for finnish", 2019.