A First Eye on the Impact of Quantum Natural Language Representations for Non-Functional Requirements Classification

FRANCESCO CASILLO, Department of Computer Science, University of Salerno, Italy
VINCENZO DEUFEMIA, Department of Computer Science, University of Salerno, Italy
FABIO PALOMBA, Department of Computer Science, University of Salerno, Italy
CARMINE GRAVINO, Department of Computer Science, University of Salerno, Italy

Classification of Non-functional Requirements (NFRs) is vital for the software development lifecycle. While various Machine Learning (ML) and Deep Learning (DL) methods have been applied to this task, they often struggle with linguistic structures and limited data.

This study explores the use of Quantum Natural Language Processing (QNLP) for NFR classification, comparing it with classical approaches. By leveraging quantum natural language representation to face the linguistic structures behind requirements, and the potential of quantum models, which are expected to handle smaller datasets efficiently, we investigate three strategies: shallow ML algorithms using word embeddings, tensor network-based models, and quantum circuit-based models. Using the PROMISE dataset, we evaluate the performance of these approaches in classifying NFRs, such as security, performance, operability, and usability.

Shallow ML methods with word embeddings like GloVe and Word2Vec achieved high performance, with F1-scores exceeding 0.80. Tensor network models using string diagrams showed promising results, while quantum models, despite their probabilistic nature, demonstrated comparable performance in some cases.

Our findings suggest that quantum approaches can compete with traditional methods in NFR classification, though pre-trained word vectors still lead in performance. This study opens new pathways for integrating quantum computing into requirements engineering, highlighting its potential and challenges.

CCS Concepts: • **Software and its engineering → Automatic programming**; • **Computing methodologies → Information extraction**; • **Theory of computation → Quantum information theory**.

Additional Key Words and Phrases: Quantum Natural Language Processing, Non-Functional Requirements, Quantum Software Engineering, Requirements Engineering, Quantum Computing, Automatic Classification, Machine Learning in Software Engineering.

ACM Reference Format:
Francesco Casillo, Vincenzo Deufemia, Fabio Palomba, and Carmine Gravino. 2024. A First Eye on the Impact of Quantum Natural Language Representations for Non-Functional Requirements Classification. In *Proceedings of TBD (Conference acronym 'XX)*. ACM, New York, NY, USA, 43 pages. https://doi.org/XXXXXXX.XXXXXXX

Authors' Contact Information: Francesco Casillo, fcasillo@unisa.it, Department of Computer Science, University of Salerno, Fisciano(SA), Italy; Vincenzo Deufemia, deufemia@unisa.it, Department of Computer Science, University of Salerno, Fisciano(SA), Italy; Fabio Palomba, fpalomba@unisa.it, Department of Computer Science, University of Salerno, Fisciano(SA), Italy; Carmine Gravino, gravino@unisa.it, Department of Computer Science, University of Salerno, Fisciano(SA), Italy.

1 Introduction

Non-Functional Requirements (NFRs) play a pivotal role in Software Engineering (SE), as they capture the system's quality attributes like security, performance, and usability, complementing functional requirements by specifying how well a the system fulfills its intended functions [2, 6]. Identifying and classifying NFRs early in the development process is crucial, since overlooking them can result in severe consequences like cost overruns, project delays, and system failures [36, 49]. In fact, numerous studies have highlighted how accurate NFR classification significantly contributes to preventing costly errors, enhancing software quality, and ensuring compliance with regulatory requirements [4, 25, 31]. Given the complexity and scale of modern software projects, automating the classification process has become increasingly attractive, offering a way to reduce human effort and error. As a result, the automatic classification of NFRs has emerged as a prominent research area, attracting growing attention within the software engineering community [15, 24, 32].

Recent advancements in Machine Learning (ML) and Natural Language Processing (NLP) have introduced effective approaches for automating the classification of NFRs. Shallow ML models, such as Support Vector Machines (SVM) and Random Forests (RF), combined with word embedding techniques like Word2Vec [34], GloVe [37], and FastText [5], have shown promising results in this task [4], often in conjunction with Deep Learning (DL) solutions, including neural networks and Transformer-based architectures [23, 48]. However, these models tend to rely heavily on pre-trained embeddings, which require large-scale corpora to acquire meaningful linguistic representations before being applied to the NFR classification domain. Moreover, they often function as black boxes, lacking interpretability and making it difficult to trace how predictions are derived, an important limitation in Requirements Engineering (RE), where transparency and explainability are critical. Despite their strong empirical performance, such models can also struggle to capture complex semantic dependencies and may perform poorly when training data is limited. Large Language Models (LLMs) have shown remarkable capabilities in various NLP tasks, including NFR classification, thanks to their extensive pre-training on diverse textual corpora. However, their application in RE also raises concerns, such as the risk of generating inaccurate or misleading information (hallucinations) and the absence of formal guarantees [13], which may affect their trustworthiness, especially in safety-critical contexts.

In the meanwhile, Quantum Natural Language Processing (QNLP) represents a promising and emerging frontier in computational linguistics and artificial intelligence, offering a novel approach to modeling language based on the principles of quantum mechanics, such as superposition and entanglement [14]. By leveraging string diagrams and tensor networks, QNLP encodes grammatical structures and word semantics directly into quantum circuits, enabling a more holistic and compositional representation of language [10, 50]. This paradigm is particularly well-suited for capturing the complex and high-dimensional linguistic dependencies often found in domains like RE. Frameworks such as the Distributional Compositional Categorical (DisCoCat) model exemplify this approach, supporting the quantum compositional representation of sentence meaning [21]. Theoretical results suggest that QNLP methods may offer significant computational advantages over classical techniques, particularly for tasks requiring the modeling of rich syntactic and semantic structures [1, 17, 47].

Unlike classical NLP models, such as BERT [11] and TF-IDF [43], which rely on surface-level statistical patterns and word co-occurrences, QNLP provides a more principled and structured approach to language understanding. In addition, QNLP models do not depend on extensive external pretraining; instead, they learn directly from task-specific data, making them inherently more adaptable to domain-specific and low-resource scenarios. Quantum properties such as superposition, entanglement, and interference enable these models to capture subtle and complex semantic

interactions in a mathematically interpretable way. These characteristics position QNLP as a promising direction not only for classification tasks but also for more advanced activities in RE, such as semantic reasoning, traceability, and conflict detection.

This paper presents an empirical study on the application of QNLP for the automatic detection of NFRs formulated as a multiclass classification task, a novel exploration in the software engineering domain. Unlike previous work that primarily emphasizes predictive performance with conventional ML/DL models and benchmark datasets [4, 25, 31], this study explores the feasibility of quantum-inspired methods. Specifically, it compares three distinct approaches: 1) traditional shallow ML models combined with word embedding techniques, serving as a baseline; 2) a tensor network-based model that uses string diagrams to represent NFRs; 3) quantum models that convert these diagrams into quantum circuits, simulating quantum computations to perform classification. QNLP, grounded in categorical quantum mechanics, offers a principled way to model the contextual and compositional nuances of natural language, making it particularly well-suited for capturing the subtle semantics inherent in NFRs, a domain that remains largely unexplored. Introducing quantum-inspired representations into such tasks demands a careful, step-by-step evaluation of their practical feasibility.

Thus, our first investigation (RQ1) aims to assess the performance of shallow ML algorithms like SVM, RF, and Linear Regression, when applied to NFR classification using word embedding techniques. In particular, unlike previous studies that rely on advanced DL models like Convolutional Neural Networks (CNNs) or BERT, our approach adopts shallow ML models, differing in terms of preprocessing strategies, data volume, and model complexity, using BERT only as tokenizer to generate embeddings for requirements. This approach paves the way for the exploration of new techniques for requirement representation, particularly through the use of string diagrams. String diagrams offer a visual and compositional framework for capturing the grammatical structure of sentences, using tensor networks to express semantic relationships between words. Representing requirements as tensor networks remains an underexplored direction in the literature, which we aim to highlight through our second research question (RQ2), focused on evaluating the impact of string diagram representations in multiclass NFR classification. Lastly, the use of string diagrams enables the integration of quantum computing techniques, as these diagrams can be systematically translated into quantum circuits. This opens the door to simulating quantum computation for the NFR classification task. Our third research question (RQ3) investigates this possibility by evaluating the effectiveness of quantum circuit representations of NFRs under various simulated quantum environments, including both shot-based and non-shot-based experimental settings.

In response to RQ1, we find that shallow ML algorithms, when combined with word embeddings like GloVe and Word2Vec, produced highly effective models for classifying NFRs. These models achieve precision, recall, and F1-score metrics exceeding 0.80, demonstrating the strength of these traditional statistical embeddings in capturing the semantic relationships in NFRs. However, embeddings such as TF-IDF and BERT, which are based on word frequency and contextual embeddings, struggle in the classification task, with significantly lower F1-scores, highlighting the difficulty of these models in handling the volume of data and the specific preprocessing steps used in this study. RQ2 explore the potential of tensor network-based models, where requirements were represented through string diagrams. The Cups Reader, which represents language as a sequence of tensors, outperforms the syntax-based Bobcat Parser, particularly when coupled with Cross Entropy loss in a PyTorch model. While the overall performance was not as high as traditional ML techniques, the results were encouraging, showing that string diagram representations can offer a competitive alternative, particularly outperforming BERT-based models in several NFR classes such as Operability, Security, and Usability. This suggests that tensor-based representations of NFRs may offer a novel path forward for requirements classification, though further refinements are necessary to improve their overall efficacy. Finally, in addressing RQ3, our experiments with quantum circuit-based models reveal a more complex landscape. The non-shot-based simulations,

which transform quantum circuits into tensor networks using a Numpy model, demonstrate comparable results to those in RQ2, even though with lower performance overall. Meanwhile, the shot-based simulations show that certain NFR classes, such as Operability, could achieve high recall, but the remaining results are generally scarce. The probabilistic nature of quantum computation and the additional factors, such as backend configuration and shot counts, introduce new complexities, making it challenging to fully evaluate the potential of quantum technologies for NFR classification in this initial exploration. Despite this, the possibility to represent NFRs through quantum circuits remains a promising direction, particularly as quantum hardware and simulation environments evolve.

In the end, the main contributions of this work are:

(1) A benchmark of combinations of word embeddings and shallow ML algorithms that differentiate from the state of the art on NFR classification task;
(2) A first attempt to classify NFRs represented as string diagrams using tensor networks.
(3) A novel application of quantum computing techniques for the NFR classification task, including the use of quantum circuits for processing linguistic structures.
(4) A road-map to further investigate the application of QNLP techniques and technologies for NFRs classification task and a broader set of RE processes;
(5) A replication package to replicate and build upon our work [8].

The remainder of this paper is organized as follows. Section 2 reviews the related work in NFR classification and quantum NLP. Section 3 introduce important concepts in QNLP that will be used in our investigation. Section 4 outlines the empirical study design and methodology. Section 5 presents the results of our experiments, followed by a discussion of findings and future research directions in Section 6. Finally, Section 7 offers concluding remarks.

2 Related Work

In this section we outline the works closest to the topic of our study. In the first section we summarize those related to the NFR classification task, discussing the approaches explored. In the second section we highlight the applications of QNLP in different fields, shedding light on how these new techniques are exploited.

2.1 State of the art on NFR Classification Task

Various studies have explored the classification of NFRs using ML and DL techniques, achieving notable successes.

Maciejauskaitė and Miliauskaitė [31] compared several ML algorithms, including Support Vector Machines, Naïve Bayes, and K-Nearest Neighbors. They developed a majority voting classifier and evaluated its performance using accuracy, precision, recall, and F1-score metrics. The majority voting classifier yielded an accuracy of 0.71, with a precision of 0.85, recall of 0.81, and F1-score of 0.82, demonstrating that a combination of ML models can lead to robust performance in NFR classification.

Li and Nong [29] proposed NFRNet, a deep neural network model for automatic NFR classification. Their approach integrates BERT-based word embeddings with a Bi-LSTM classifier, leveraging N-gram masking to improve context representation. They introduced multi-sample dropout regularization to enhance generalization and reduce training time. The model was evaluated on an expanded dataset (PROMISE + SOFTWARE NFR) and outperformed 17 baseline methods. Their best model achieved an F1-score of 0.91, with a precision of 0.89 and a recall of 0.92, demonstrating the effectiveness of deep learning in NFR classification.

Kaur et al. [23] proposed a hybrid model combining the BERT pre-trained language model with a CNN. The BERT model captures contextual language features, while the CNN layer enhances the classification performance by improving feature extraction. The experiment was conducted on the PROMISE dataset. The BERT-CNN model outperformed baseline methods, providing better accuracy for NFR classification, thanks to the powerful feature extraction and contextual understanding provided by BERT.

Khayashi et al. [25] evaluated five DL algorithms for classifying functional and non-functional requirements from the PURE dataset. They also implemented ensemble learning techniques, such as voting classifiers, to further improve classification performance. The ensemble methods, which combined multiple DL models, provided strong classification performance, indicating that DL models and ensemble techniques are effective for large-scale NFR classification tasks.

Rahman et al. [41] conducted a comparative analysis of 15 ML algorithms and various feature extraction techniques, including Bag of Words (BoW), Term Frequency and Inverse Document Frequency (TF-IDF), and Chi-Squared vectorization. They examined which combination of classifiers and feature extraction methods provided the best results. The Linear SVC classifier combined with TF-IDF vectorization achieved the best results, with an F1-score of 0.82, highlighting the importance of feature extraction techniques in NFR classification.

Table 1. Summary of approaches on NFR classification.

Reference	Approach	Dataset Used	NFRs Targeted	Results Obtained
[31]	Majority voting classifier (SVM, Naïve Bayes, KNN)	Custom dataset (industry requirements)	Performance, security, usability, maintainability, scalability	Accuracy: 0.71, Precision: 0.85, Recall: 0.81, F1-score: 0.82
[29]	NFRNet (BERT + Bi-LSTM) with multi-sample dropout	PROMISE + SOFTWARE NFR dataset (expanded)	Availability, security, maintainability, usability, performance, portability	F1-score: 0.91, Precision: 0.89, Recall: 0.92
[23]	BERT-CNN (BERT with convolutional layers)	PROMISE dataset (625 requirements)	Usability, security, operational, performance	Accuracy: 0.94, Recall: 0.97, F1-score: 0.92
[25]	Deep Learning (CNN, LSTM, GRU) + ensemble voting	PURE dataset (4661 requirements)	Availability, security, efficiency, fault tolerance, usability, scalability	F1-score: 0.88, Accuracy: 0.92
[41]	ML algorithms (SVM, TF-IDF, Chi-Squared)	PROMISE_exp dataset	Security, availability, performance	F1-score: 0.82, Accuracy: 0.85

In contrast to these prior works, which mainly rely on classical ML and DL techniques, our approach introduces the use of quantum circuits for NFR classification. By leveraging quantum computing's unique properties, such as superposition and entanglement, we explore a fundamentally new computational paradigm that diverges from traditional methods. Our work utilizes tensor networks and quantum circuits based on string diagrams, which opens new avenues for addressing classification tasks in the field of software engineering and, more in general, in NLP tasks. Such approach represents a total novel contribution to the literature in SE, as quantum computing hold the potential to provide computational advantages for large-scale and high-dimensional problems. Despite this potential, being unexplored territory, the idea of our work is to provide an initial empirical study and understand how this approach is positioned in relation to widely explored strategies in the field.

2.2 State of the art on QNLP applications

This section reviews the state of the art in QNLP approaches, highlighting key developments and applications in various domains.

Maha A. Metawei et al. [33] explore a hybrid quantum-classical model for topic classification. The QNLP model is used to determine if two sentences are related to the same topic or not. They compare QNLP models to classical tensor network-based models, showing significant improvements in training (0.45) and validation accuracy (0.35). In this work the authors analyzed the convergence of the QNLP model by varying the problem size, the parameterized quantum circuits used for training, and the noise model of the backend quantum simulator, with the result that strongly entangled ansatz has the fastest model convergence.

Kundu et al. [27] apply quantum tensor networks for protein classification. They treat protein sequences like sentences in NLP and map them into parameterized quantum circuits, using Quantum Tensor Networks architectures similar to classical RNNs and CNNs. Their top-performing quantum model achieved 0.94 accuracy, comparable to classical models, but with significantly fewer parameters.

Vinay R. and Badari Nath K. [40] investigate quantum video classification by transforming video captions into quantum circuits. Their method reduces video classification to a quantum text classification problem and feeds quantum circuits into a classifier. Their proposed quantum video classifier achieved 0.89 accuracy on the Kinetics dataset.

Herrington et al. [17] explore quantum computing's potential in medical robotic surgery and drug discovery. The study outlines the integration of QNLP to enable robots to better understand and communicate in human languages for improved robotic surgery outcomes. The study presents theoretical improvements in communication between robots and humans using QNLP but lacks detailed experimental results.

Abbaszade et al. [1] explore quantum machine translation using QNLP algorithms on noisy intermediate-scale quantum (NISQ) devices. They propose an encoder-decoder model with quantum circuits and employ Shannon entropy to enhance translation between English and Persian. The best model achieved a mean absolute error of 0.03 and mean squared error of 0.002 using two LSTM layers with optimized quantum circuits.

Stein et al. [47] apply QNLP to sentiment analysis in finance. They compare two quantum approaches, DisCoCat and Quantum-Enhanced LSTM (QLSTM), and generate financial datasets using a ChatGPT-based approach for case studies. QLSTM models performed faster in training and achieved results close to classical models, making them promising candidates for sentiment analysis tasks in finance.

Our work differs from these studies by specifically addressing the classification of NFRs using both classical and quantum approaches. While other studies apply QNLP techniques to a variety of NLP tasks (e.g., topic classification, sentiment analysis, machine translation), our focus is on the intersection of quantum circuits and software requirements engineering. We introduce the exploration of tensor networks and quantum circuits, using tools like Bobcat Parser and Cups Reader, better detailed in the following sections, which represent a unique application of quantum techniques to NFR classification in software development. This combination of classical and quantum models for NFR classification has not yet been explored in depth, setting our research apart from existing literature.

3 Introduction to Quantum NLP

This section provides a conceptual foundation for the QNLP methods employed in our empirical study. While not all models or options described here are used in our experiments, they represent the broader design space made available through tools like lambeq. Understanding this space is essential to understand our methodological choices, which are

explained in detail in Section 4. In particular, we selected specific combinations, i.e., the Bobcat parser and Cups Reader for string diagram generation, the Tensor Ansatz for classical simulation, and the IQP Ansatz for quantum circuits, based on their alignment with the compositional goals of our study and their feasibility given current simulation constraints. These choices are motivated by the need to evaluate the potential of compositional and quantum-inspired representations in NFR classification, using both classical and simulated quantum setups.

Quantum computing. Quantum computing is an emerging paradigm that leverages the principles of quantum mechanics to perform computations that are infeasible for classical computers. Unlike classical bits, which can be either 0 or 1, quantum bits (qubits) exist in a superposition of both states simultaneously. This property, combined with entanglement—a phenomenon where qubits become interdependent regardless of distance—enables quantum systems to process information in fundamentally novel ways.

A quantum computation is performed using a sequence of quantum gates, which are unitary operations that manipulate qubits in a reversible manner. For example, Rz, Rx and Ry gates are phase rotations around the Z-axis, X-axis and Y-axis of the Bloch sphere. These are just a few of the unitary operators, i.e. general quantum transformations that preserve probability amplitudes. Such operators can be linked to define an ansatz (plural: ansätze), a structured architecture design with adjustable parameters, often used in variational quantum algorithms to approximate solutions.

Quantum circuits are typically designed using quantum circuit ansätze, where parameters are trained using optimization techniques to achieve specific computational objectives.

QNLP. QNLP is a developing field that applies quantum computing principles to the challenges of NLP. This field aims to leverage the unique capabilities of quantum computers—such as superposition, entanglement, and quantum parallelism—to enhance the performance of NLP tasks, hopefully surpassing the limits of classical computational methods. Traditional NLP models, such as word embeddings (e.g., Word2vec [34]) and deep learning architectures like Transformers [48], have revolutionized the processing of human language. However, these models are inherently bound by the constraints of classical computation. They typically represent words as vectors in a high-dimensional space and apply operations that, while effective, are limited in capturing the full complexity of a language with its ambiguity.

QNLP introduces a new paradigm by representing linguistic elements as quantum states. In this framework, words, phrases, and sentences are encoded into quantum systems, where the principles of quantum mechanics allow for a more expressive representation of linguistic meaning: for example, superposition enables the simultaneous representation of multiple meanings or interpretations of a word, while entanglement allows for the complex dependencies between words to be captured more naturally than in classical models. One of the foundational models in QNLP is the Distributional Compositional Categorical (DisCoCat) model, which combines category theory with distributional semantics to model the meaning of sentences in a compositional manner [10]. This model provides a natural fit for quantum computing, as it aligns with the structure of quantum mechanics, where compositionality plays a key role.

To facilitate the development and experimentation of QNLP models, the lambeq library [22] was created by the Quantinuum QNLP team[1]. Lambeq is an open-source Python framework that provides functionality for converting any sentence into a quantum circuit, utilizing a specified compositional model along with particular parameterizations and selected ansätze. The process of converting a sentence into a quantum circuit involves several key steps, each offering a variety of functionalities. These steps include parsing and encoding, rewriting, parameterisation, and training/testing, as depicted in Figure 1. We summarize the various features provided from the framework. The full description of each

[1] https://www.quantinuum.com/

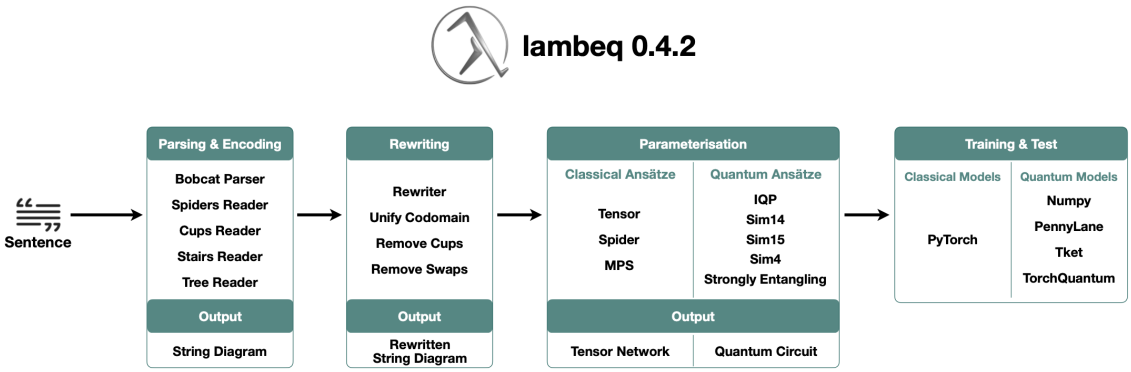


Fig. 1. The general pipeline implemented through lambeq.

can be found in the specific documentation[2], from which we grasped most of the following information and in which can be found visual example for each kind of parsing, rewriting and parameterisation.

Parsing & Encoding. The parsing process is the step that converts a sentence into a string diagram. Lambeq's string diagrams are equipped with types that represent the interactions between words according to the pregroup grammar formalism [28]. When annotated with pregroup types, the diagram for a sentence shows each wire labeled with an atomic type or an adjoint. For instance, in the sentence "John walks in the park", the noun "John" might be labeled with type n, corresponding to a noun or noun phrase, while the verb "walks" could be labeled with type s, corresponding to a sentence. The adjoints n^r and n^l indicate that a noun is expected to the right or left of a specific word, respectively. The library provides different syntactic models, offering flexibility in how linguistic structures are represented and processed. In the following we detail the various parsing methods supported by lambeq, with a focus on the DisCoCat model and alternative approaches such as bag-of-words, word-sequence models, and tree readers.

The DisCoCat model is central to lambeq's syntax-based parsing capabilities. This model combines category theory with distributional semantics to represent sentence meaning compositionally, generating a Combinatory Categorial Grammar (CCG) string diagram.

The Spiders reader is an example of a bag-of-words approach where a sentence is represented as a collection of words without regard to order, combining the words using a spider, which is a commutative operation. In this case, the diagram does not follow pregroup grammar rules, as the model treats words as unordered elements, which may be suitable for tasks where word order is less significant.

The Cups Reader and Stairs Reader models are for sequential word composition. These models compose words from left to right in a linear sequence. In particular, the Cups Reader generates a "tensor train", ensuring that the final sentence representation is an order-1 tensor. A START symbol, represented as an order-1 tensor (a vector), is used at the beginning of the sentence to maintain this structure.

The Stairs Reader uses a recurrent structure, similar to a recurrent neural network (RNN), combining consecutive words through a "cell" represented by a box.

Tree Reader provide another syntax-based parsing approach, leveraging the structure of CCG derivations. A CCG derivation, which follows a biclosed form, is directly interpreted as a sequence of compositions. By default, composition

[2] https://docs.quantinuum.com/lambeq/intro.html

is handled by a single "cell" named UNIBOX, even though lambeq offers customization through the TreeReaderMode, allowing for different cells based on CCG rules or rule-type pairs and enabling more fine-grained control over the composition process.

Rewriting. The rewriting stage is adopted for simplifying and optimizing string diagrams derived from syntactic derivations. These diagrams, which represent the structure and relationships between words in a sentence, can become complex, leading to increased computational costs and longer training times. The rewriting process aims to address these issues by applying various rewriters and rewriting rules to reduce the complexity of the diagrams. The rewriting can be applied at two stages: box-level and diagram-level.

The former apply transformations at the level of individual boxes within a diagram. Each rewriting rule operates on one box at a time, with no access to the broader context of the diagram. This localized approach is useful for making specific, targeted changes to parts of the diagram without considering the entire structure. The rewrite module provides several rules for simplifying diagrams. For example, the "prepositional_phrase" rule reduces the complexity of prepositional phrases by using a "cap" to bridge discontinuous wires, effectively lowering the order of the tensor associated with the preposition. The determiner rule, on the other hand, eliminates determiners like "the" by applying a cap on the corresponding type, simplifying noun phrases.

Unlike box-level rewriters, diagram-level rewriters operate on the entire diagram, enabling more comprehensive transformations that require an understanding of the broader context within the diagram. There are tree main diagram-level rewriters: Unify Codomain Rewriter, Remove Cups Rewriter and Remove Swaps Rewriter.

The Unify Codomain Rewriter is particularly useful when dealing with diagrams that have multiple free wires, which could lead to dimension mismatch errors during training. The Unify Codomain Rewriter merges these free wires into a single wire, ensuring consistent training. It does so by looking at the codomain of the diagram and combining multiple wires into one using an additional box.

The Remove Cups Rewriter removes cups from a diagram, reducing or eliminating post-selection. By doing so, it streamlines the diagram and enhances computational efficiency, making it easier to implement on quantum hardware.

The Remove Swaps Rewriter eliminates swaps from a diagram, producing a proper pregroup diagram that aligns with J. Lambek's definition of pregroup grammars [28]. This rewriter is essential for maintaining the structural integrity of the diagram in accordance with the theoretical underpinnings of pregroup grammar.

Parameterisation. The parameterisation phase consists in transforming the string diagrams derived from sentences into concrete quantum circuits or tensor networks. This stage involves the application of ansätze, which define the specific parameters for the quantum or classical systems, such as the number of qubits or tensor dimensions associated with each word or grammatical type in the sentence. In classical experiments, Tensor, Matrix Product States and Spider Ansatz can be applied to convert the string diagram into a tensor network. This involves assigning dimensions to the atomic types in the diagram, defining how words or grammatical types are represented as tensors.

In particular, the Tensor Ansatz assigns specific dimensions to the noun and sentence spaces, creating a tensor network where each wire in the diagram corresponds to a specific dimension. This approach is suitable for classical NLP tasks where quantum resources are not available. The Matrix Product States (MPS) Ansatz is used to convert large tensors, which can become unwieldy in classical experiments, into sequences of smaller, order-3 tensors connected with cups. This approach is necessary when tensors are too large to be efficiently processed. The user have to also define the bond dimension, which specifies the dimensionality of the wires connecting these tensors. The Spider Ansatz is another method for managing large tensors, where tensors of order greater than 2 are split into sequences of order-2 tensors

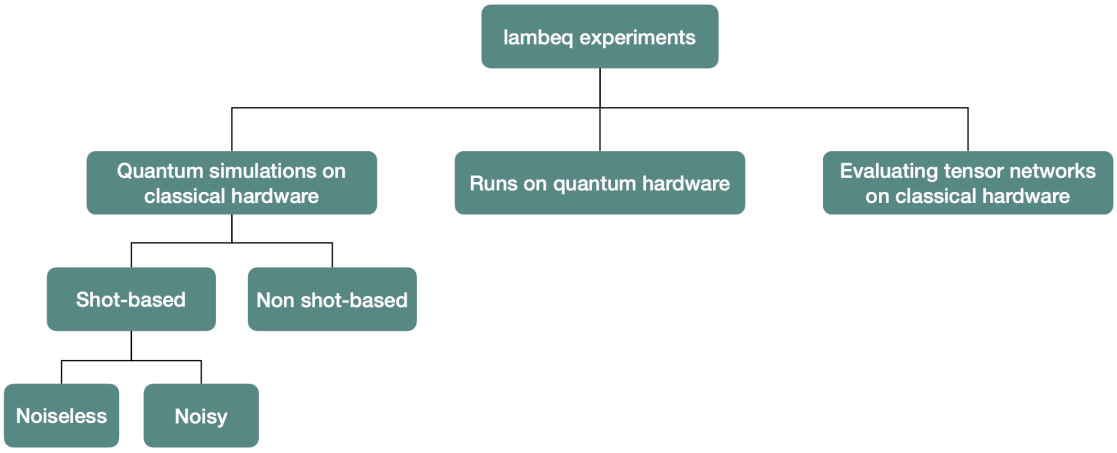


Fig. 2. Hierarchy of experimental applications in lambeq.

(matrices) connected with spiders. This method reduces the space required to store tensors and makes the computation more tractable.

In quantum experiments, the string diagram is converted into a quantum circuit by applying a Circuit Ansatz. Lambeq provides several quantum ansätze, each with different configurations and properties suitable for various quantum computing tasks. The Instantaneous Quantum Polynomial (IQP) Ansatz interleaves layers of Hadamard gates with diagonal unitaries. The diagonal unitaries are implemented using adjacent controlled-Rz (CRz) gates [16]. This ansatz is particularly useful for creating circuits that are classically hard to simulate. To convert a string diagram into an IQP circuit, you assign qubits to atomic types (e.g., 1 qubit for nouns and 1 qubit for sentences) and specify the number of IQP layers. Sim14 Ansatz is a modification of Circuit 14 define by Sim et al. [45], where the circuit-block construction is replaced with two rings of CRx gates, oriented in opposite directions. It provides a different entanglement structure compared to the IQPAnsatz. Sim15 Ansatz is similar to Sim14Ansatz, but uses two rings of CNOT gates instead of CRx gates, providing another variation in the entanglement structure. Sim4 Ansatz corresponds to Circuit 4 defined as Circuit 14 by Sim et al. [45]. It employs a layer of Rx gates followed by Rz gates, and a ladder of CRx gates per layer, making it a more straightforward but effective quantum circuit design. Adapted from the PennyLane implementation, Strongly Entangling Ansatz uses three single-qubit rotations (RzRyRz) followed by a ladder of CNOT gates with varying ranges per layer. This ansatz is designed to create strong entanglement across the qubits, which is crucial for certain quantum algorithms.

Training & Test. The training and testing stage involves using quantum circuits or tensor networks obtained from the parameterisation to train models for various machine learning tasks. Lambeq provides several models, each suited to either quantum or classical experiments.

All the possible experiments that can be done through lambeq are shown in Figure 2.

The model employed to evaluate tensor networks on classical hardware is the Pytorch Model. It treats string diagrams as tensor networks, where each box in the diagram corresponds to a tensor, and the edges between boxes define how these tensors are contracted (i.e., how the data flows between different parts of the model). The tensor contractions in

this model are optimized using opt_einsum, a library that efficiently handles complex tensor operations. This makes the Pytorch Model capable of handling large and intricate tensor networks typical in advanced machine learning tasks. The Pytorch Model is versatile and can be integrated with classical neural networks. Users can subclass the model and override the forward method to combine the outputs of tensor diagrams with other neural network layers, allowing for the creation of complex and customized architectures.

The Numpy Model is a quantum model that utilizes low-level simulators to convert quantum circuits into tensor networks. These tensor networks are then contracted efficiently using the opt_einsum library. The model supports two types of simulators: unitary simulators and density matrix simulators. The Unitary Simulator is used when the circuit contains only Bra, Ket, and unitary gates. It converts the circuit with n output qubits into a tensor of shape (2,) * n, which is a pure quantum state. On the other hand, the Density Matrix Simulator is used when the circuit involves operations like Encode, Measure, or Discard, which result in a mixed quantum state. It converts the circuit into a tensor of shape (2,) * (2 * n + m), where n is the number of output qubits and m is the number of output bits. This allow the exact (non shot-based) simulation of quantum circuits on classical hardware. A significant feature of the Numpy Model is its ability to use just-in-time (jit) compilation provided by the jax library. This dramatically speeds up the evaluation of the model by pre-compiling functions, making it ideal for rapid experimentation. However, using jit compilation requires substantial memory, so it is best suited for smaller models.

The PennyLane Model leverages the capabilities of PennyLane[3], a popular quantum machine learning library, along with PyTorch, to create hybrid quantum-classical models. This model allows to perform classical-quantum machine learning experiments where quantum circuits are combined with classical neural networks. The model offers two modes of operation: State Vector Simulation and Probability Simulation. The former mode simulates the quantum circuit as a pure quantum state using unitary operations. It is efficient for simulation purposes but does not account for measurement noise. The latter mode simulates the quantum circuit as a mixed state, taking into account the probabilities of different outcomes after measurements. This mode is required when running experiments on real quantum hardware, where measurement noise and errors are present. The PennyLane Model can be used to optimize simulated circuits using exact back-propagation in PyTorch, which can lead to more accurate training compared to optimization methods like Simultaneous Perturbation Stochastic Approximation (SPSA) [46]. However, exact back-propagation is not feasible on real quantum hardware, where the parameter-shift rule is used instead. Additionally, the model supports hybrid architectures where the output of a quantum circuit is fed into a classical neural network. This allows for the creation of complex models that leverage both quantum and classical computational advantages.

The Tket Model is designed for running quantum circuits on real quantum hardware or in simulated environments that account for hardware-specific noise and architecture constraints, allowing quantum simulations in noisy shot-based environment. This model utilizes the “pytket” library, a high-performance simulator framework that is a part of the Qiskit suite, developed by IBM [39], which provides access to various quantum hardware backends. This framework supports a variety of backends, including AerBackend (the one that we used in our experiments) for noisy simulations of quantum machines and direct execution on devices like Quantinuum’s H-series or IBM Quantum hardware. The Tket Model is particularly useful for experiments that require shot-based results, which are the counts of measurement outcomes after running a quantum circuit multiple times. These shot-based results are essential for understanding the probabilistic nature of quantum computing. The model can be configured to use different noise models and compilation

[3]https://pennylane.ai/

passes specific to the hardware, making it a versatile choice for experiments that need to consider the limitations and characteristics of actual quantum devices.

The TorchQuantum Model is designed for running exact non-shot-based simulations of quantum circuits. It uses torchquantum, a statevector-based simulator that integrates with PyTorch, allowing for precise gradient-based optimization of quantum circuits. This model is particularly advantageous when precise simulations are required without the need for probabilistic outcomes, which are typical in shot-based quantum computing. The TorchQuantum Model supports exact back-propagation, making it suitable for tasks where accurate gradient computation is crucial. However, this exactness comes at the cost of not being applicable to real quantum hardware, where the probabilistic nature of quantum operations must be considered.

4 Empirical Study Design

This section provides an overview of the empirical study, starting with the reasoning behind the formulation of our research questions. We then present the dataset selected for our experiments, including the data sources and preprocessing methods. After, we outline the evaluation metrics employed to measure the predictive accuracy of the Quantum NLP approaches under consideration. Lastly, we address potential threats to the validity of our study and describe the strategies implemented to minimize these risks.

In this study, we aim to investigate how Quantum NLP strategies and models can be applied to categorize non-functional requirements and compare their performance against classical NLP approaches. We aim to understand the specific advantages, if any, Quantum NLP may offer in handling such task and to evaluate whether the progress on this new field translate into measurable improvements in classification efficiency.

Our Goal. Explore the potential of Quantum NLP in the context of non-functional requirements classification.

4.1 Motivations of Research Questions

In Section 2, we discussed how different works have exploited ML and DL to address the multiclass NFRs classification problem. However, each of them used a different dataset, focusing on specific NFRs, different preprocessing techniques, different word embedding methods, and different validation techniques. Since in this work we introduce a new way of representing requirements to prepare them later for training ML models, we felt it was essential to conduct an initial exploration that would allow us to make a comparison, based on the same dataset, the same metrics, and using the same validation method. To evaluate the effectiveness of Quantum NLP in comparison to traditional NLP methods, we begin by determining whether shallow ML models combined with traditional word embedding techniques are suitable for the specific task. This consideration led to the formulation of our first research question:

RQ1. How effectively can shallow machine learning algorithms classify non-functional requirements when represented with word embeddings techniques?

By answering RQ1 we have the baseline for the comparison against Quantum NLP strategies. Since we have different choices in this new approach (see Figure 2), as described in Section 3, we start our investigation by evaluating tensor networks on classical hardware, exploiting the classical path in the pipeline shown in Figure 1. This choice is due to the fact that first and foremost we want to understand how the representation of requirements by string diagrams impacts the performance of the models, without changing the underlying computational model. This also allows us to

make a comparison with classical methods (word embeddings and shallow ML) that exploit a different representation of requirements to predict their class. This lead to the definition of the following research question:

> **RQ2.** How effectively can a basic model classify non-functional requirements when represented as string diagrams and parameterised as tensor networks?

Given the novelty of string diagrams and tensor networks in representing language structures, this research question seeks to examine the effectiveness of symbolic and sequential parsing models in classifying NFRs. The focus is to assess the capability of tensor networks, which offer a distinct and mathematically elegant representation of language, to provide more accurate classifications compared to traditional word embeddings. The representation in string diagrams opens the road for our last investigation that cover the quantum simulation on classical hardware, leading to the formulation of the following research question:

> **RQ3.** How effectively can quantum models classify non-functional requirements when represented as string diagrams and parameterised as quantum circuits?

Quantum computing offers the potential to enhance problem-solving capabilities for specific classes of computational tasks. In this context, our third research question (RQ3) investigates whether quantum circuits, representing language as tensors, can lead to better or comparable classification performance to classical machine learning and tensor-based models. Given the inherently probabilistic nature of quantum computation and the complexity introduced by factors such as shot counts and compilation passes, this investigation explores whether quantum-enhanced models can provide a tangible advantage in this domain or whether they introduce new challenges, such as parameterization and circuit complexity. Due to the current unavailability and limitations of practical quantum hardware, we simulate a quantum environment to conduct our experiments, as illustrated in Figure 2. This decision is driven by several considerations. First, access to real quantum devices remains restricted, costly, and subject to significant job queuing times. High-quality quantum computing platforms are typically available only through paid cloud services, making large-scale experimentation prohibitively expensive. Furthermore, today's quantum devices are still in their infancy, with constraints such as limited qubit counts, short coherence times, high gate error rates, and shallow circuit depth capabilities. Even when hardware access is granted, effective use often requires complex error mitigation strategies, adding an additional layer of implementation and analysis complexity. Simulating quantum circuits on classical hardware, while not free of limitations, offers a practical and controlled alternative for early-stage experimentation. However, classical simulations of quantum systems scale exponentially with the number of qubits, rendering large-scale quantum model testing infeasible. To strike a balance between theoretical validity and practical relevance, we adopt two complementary simulation strategies: non-shot-based and shot-based simulations. Non-shot-based simulations are exact and noiseless, meaning they do not introduce any of the hardware noise present in real quantum devices. They are ideal for exploring the theoretical properties of QNLP, but they do not reflect real-world execution conditions. Shot-based noisy simulations, instead, incorporate simulated hardware noise models, making them more representative of actual quantum hardware. While they still operate on classical hardware, they introduce realistic noise factors such as qubit decoherence, gate errors, and circuit depth constraints. This approach provides insights into how QNLP models might behave in real-world quantum execution environments.

Our findings from these simulations serve as an initial assessment of QNLP's potential in RE. Future research should explore execution on real quantum hardware as technology advances and devices become more accessible and reliable.

4.2 Dataset

To answer the research questions presented in the previous Section 4.1, we exploit the PROMISE expanded dataset [30]. We choose this dataset because previous studies exploited it [41, 42, 44] to deal with the NFRs classification task.

It includes in total 969 requirements detailed in Table 2 coming from 15 different projects. However, we do not use the entire dataset because the conversions of requirements into string diagrams and, then, into tensor networks or quantum circuits could be hard to handle for long requirements. Furthermore, utilizing a larger dataset would demand significantly more hardware resources, particularly in quantum simulations. Consequently, we implemented various strategies to render the requirements manageable for our experiments. While these data manipulation techniques allowed us to address our research questions, they influenced with no doubt the outcomes of our experiments.

Firstly, we reduced the number of requirements from 969 to 354 by selecting only on the most frequent non-functional requirements, specifically security (SE), usability (US), operational (O), and performance (PE) requirements. Then, we apply some text normalization techniques. In particular, we simplify the text by eliminating commas, periods, question marks, and other punctuation marks. We remove common English words (e.g., "and", "the", "is") that usually don't add much value or information to the overall meaning of a sentence. We apply the lemmatization technique, reducing words to their base or root. For example, "running" would be lemmatized to "run", and thus reducing the vocabulary size to group similar words together. We calculate the mean size, in terms of words, of requirements in the dataset at this point, and it is a little bit more than 22 words per each requirement. To align with the constraints of quantum simulation, we filter out requirements exceeding 15 words, resulting in a refined subset of 282 requirements. This decision is motivated by the fact that QNLP-based models rely on quantum circuits whose complexity, and consequently, qubit count and computational cost, increases with sentence length. Simulating large circuits becomes impractical, making comparisons with classical NLP methods unreliable. To ensure a fair evaluation, we adapt the dataset to fit current quantum simulation limits, allowing for low-depth circuits that closely approximate real quantum execution. Lastly, to handle class imbalance in the dataset we randomly reduce the number of samples in the majority class so that all classes have the same number of samples. By balancing the dataset, the intention is to help prevent bias in the model towards the majority class and improve overall performance in multi-class classification tasks. We end our preprocessing with a total of 216 requirements, 54 per each NFR considered.

Finally, to use the same requirements set in the different experiment scenario, we divide the 216 requirements in train, test and validation test in percentage of 0.6, 0.2 and 0.2, respectively. In the end, the train set counts 128 requirements, the test set counts 44 requirements and the validation set counts 44 requirements.

4.3 Research Methodology

In this section we outline the methodology we apply to answer to each research question we define.

4.3.1 ***Study design to answer RQ1.*** RQ1 aims to determine the effectiveness of various combinations of word embeddings and shallow ML algorithms for the task of classifying NFRs. The approach is illustrated in Figure 3.

To identify the most effective approach, we experiment with multiple embedding techniques followed by a set of ML algorithms. Specifically, the study examines five distinct word embedding techniques, namely TF-IDF, Word2Vec, GloVe, FastText, and BERT, each selected for its unique characteristics and previous applications in similar contexts [12].

TF-IDF [43] is employed for its simplicity and effectiveness in extracting informative features from textual data. It is particularly useful in scenarios where interpretability of word importance is crucial. Word2Vec [**?**] is included for its ability to capture semantic relationships between words by learning dense vector representations, making it

Table 2. Details of the PROMISE expanded dataset.

Class	Description	Frequency
Availability (A)	The likelihood that the system is accessible to users at any given time.	31
Fault Tolerance (FT)	The capability of a system, product, or component to continue operating as intended despite the occurrence of hardware or software faults.	18
Legal & Licensing (L)	The necessary certificates or licenses required for the system.	15
Look & Feel (LF)	The aesthetic style and appearance of the product.	49
Maintainability (MN)	The effectiveness and efficiency with which the product or system can be modified by its maintainers.	24
Operability (O)	The ease with which a product or system can be operated and controlled.	77
Performance (PE)	The performance of the system relative to the resources used under specified conditions.	67
Portability (PO)	The effectiveness and efficiency with which a system, product, or component can be transferred to different hardware, software, or usage environments.	12
Scalability (SC)	The capability of a product or system to be adapted effectively and efficiently for different or evolving hardware, software, or operational environments.	22
Security (SE)	The ability of a system or product to protect information and data, ensuring appropriate access levels based on user authorization.	125
Usability (US)	The extent to which a system can be used by specified users to achieve specific goals with efficiency, effectiveness, and satisfaction in a context.	85
Functional (F)	A general category encompassing functional requirements.	444
Total Requirements		**969**

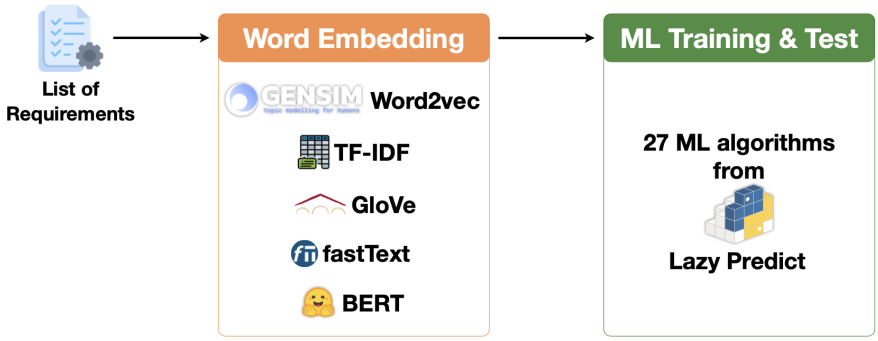


Fig. 3. Process applied to address RQ1.

suitable for tasks requiring an understanding of word meaning beyond simple word counts. Specifically, we utilize pre-trained vectors derived from a subset of the Google News dataset with nearly 100 billion words. The model comprises 300-dimensional vectors for 3 million words and phrases, with the latter generated using a data-driven method described in [35]. GloVe [37] is chosen for its capability to model global word co-occurrence statistics, which helps in capturing the semantic structure of the text at a higher level. For this we use pre-trained glove vectors based on Wikipedia 2014 plus Gigaword 5 (6B tokens, uncased). FastText [5, 19, 20] is exploited for its proficiency in handling sub-word information, making it ideal for processing languages with rich morphological variations. For our experiment, we

exploit the 1 million word vectors trained with subword infomation on Wikipedia 2017, UMBC webBase corpus and "statmt.org" news dataset (16B tokens). Lastly, BERT [11], a transformer-based model, is employed for its advanced contextual embeddings, which capture bidirectional context, thus enabling more nuanced understanding of the text, particularly in complex NLP tasks. Context-free models like Word2vec and GloVe produce a single "word embedding" for each word in the vocabulary. As an example, a word like "table" would have the same representation in both "a book on the table" and "a table in the book". In contrast, contextual models create a unique representation for each word, depending on the surrounding words in the sentence. In our work, we only leverage the Bert Tokenizer, in particular the "bert-base-uncased", to embed our requirements.

To determine the optimal ML model for classifying non-functional requirements, we utilize the *Lazy Predict*[4] library, which offers a streamlined method for comparing a broad range of classifiers without requiring extensive manual tuning of hyperparameters. This empirical investigation involves testing 27 different ML algorithms provided by *Lazy Predict*, applied to the outputs of the five embedding techniques mentioned above.

It is worth noting that in our study, we use a collection of requirements coming from different projects [30]. Details about the datasets can be found in Section 4.2. This have been chosen because previous studies exploited them [41, 42, 44], although each of the papers exploits the data for different purposes, such as including all NFRs instead of a particular set, and, thus, in different experimental setups.

To assess which is the best combination we perform a hold-out validation, dividing the dataset in train, test and validation set and calculating the metrics listed in Section 4.4. So, we firstly take the best model, based on the Accuracy and the F1-score, by using Lazy Predict for each Word Embedding. Then we test it on the hold out set to assess its performance on unseen requirements.

4.3.2 ***Study design to answer RQ2.*** The second research question aims to explore the effectiveness of classical models based on tensor networks for the task of classifying NFRs. To address this, we employ string diagrams, which are well-suited for capturing the compositional structure of language in a mathematical way. The approach begins by representing requirements as string diagrams, which are then parameterized into tensor networks. These tensor networks serve as input for classical ML models that are tasked with the classification of NFRs. The entire process involves several steps, each leveraging specific techniques and tools specified in Figure 4.

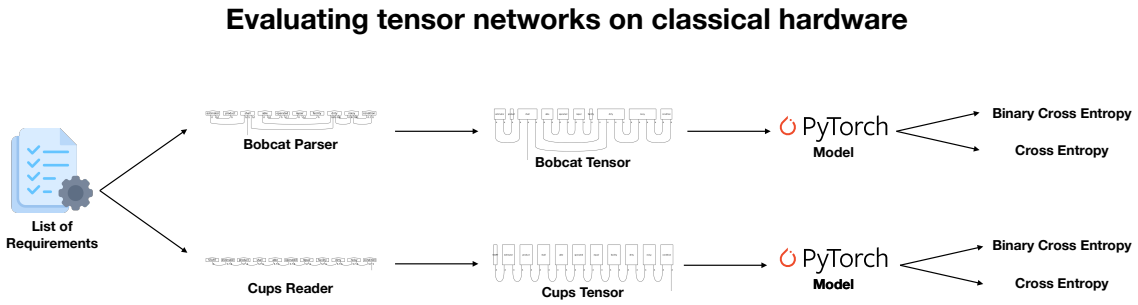
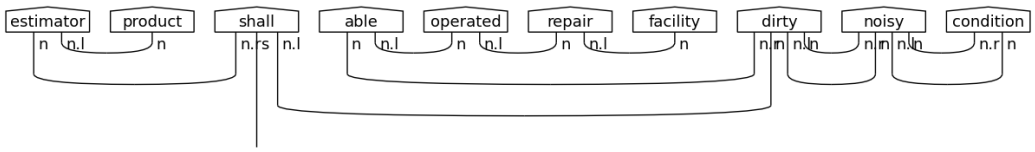


Fig. 4. Process applied to address RQ2.

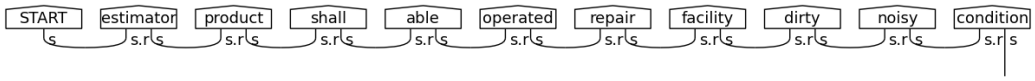
Initially, requirements are parsed and represented as string diagrams using a compositional model. In our study, we utilize the DisCoCat (Distributional Compositional Categorical) model, which is known for its ability to combine syntax

[4]The *Lazy Predict* library: https://github.com/shankarpandala/lazypredict.

(from categorical grammars) with semantics (from vector spaces). The string diagrams are created using Bobcat parser, a state-of-the-art statistical CCG parser [9], and Cups Reader. An example output is given in Figure 5. The former converts the syntactic structure of the sentence into a pregroup diagram (5a), the latter combines words linearly using a cup diagram (5b). When annotated with pregroup types, the diagram for a sentence shows each wire labeled with an atomic type or an adjoint. For instance, a word might be labeled with type n, corresponding to a noun or noun phrase, while the verb could be labeled with type s, corresponding to a sentence. The adjoints n^r and n^l indicate that a noun is expected to the right or left of a specific word, respectively.



(a) Requirement parsed with Bobcat Parser.

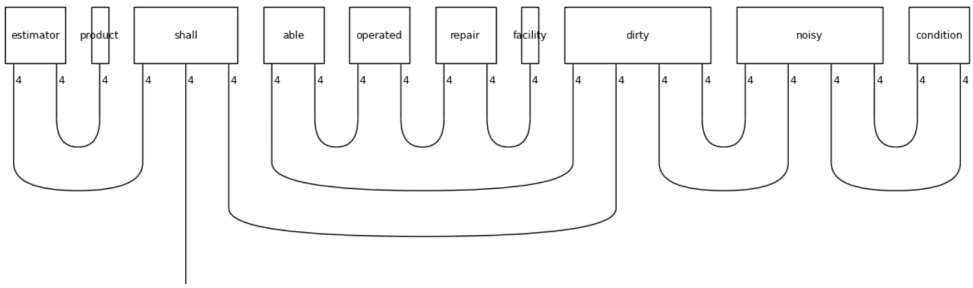


(b) Requirement parsed with Cups Reader.

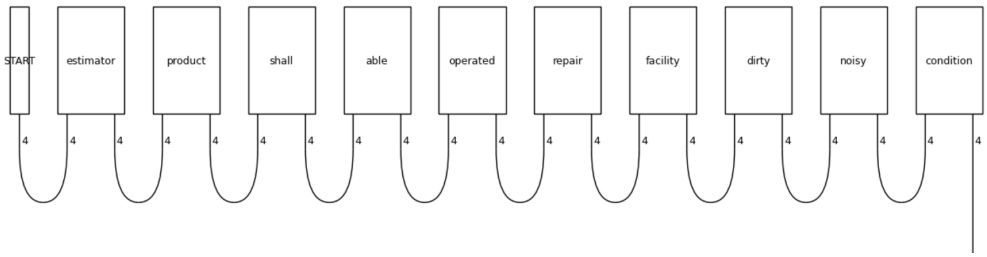
Fig. 5. Example of requirements converted in string diagram.

Once the string diagrams are constructed, the next step involves parameterizing these diagrams into tensor networks. This is achieved through the application of a Tensor Ansatz. In this study, we employ a classical Tensor Ansatz, which assigns specific dimensions to the atomic types in the string diagram, effectively converting the abstract diagram into a concrete tensor network. In our experimentation, we assign "4" as dimension to both the atomic types of "sentence" and "noun", as shown in Figure 6. The choice of dimensions is guided by the linguistic properties of the words and the grammatical types they are associated with. The tensor networks derived from the string diagrams are then contracted by the model to obtain the final classification. Tensor contraction is a process that involves summing over shared indices in the network, resulting in a lower-dimensional representation that encapsulates the meaning of the entire sentence. Examples of tensor networks are given in Figure 6, where Figure 6a is the parameterized version of the string diagram coming from Bobcat Parser, while 6b is the parameterized version of the string diagram coming from Cups Reader. These tensors are then fed into a basic ML model which is trained to classify the requirement as belonging to different categories of non-functional requirements.

The classical ML model is trained on a labeled dataset of non-functional requirements, where each requirement has been pre-categorized. In this experiment, we exploit the PyTorch model provided by lambeq. It is possible of combining tensor networks and neural network architectures to it, however, we use its default definition as provided by the library. As usually, the training process involves optimizing the model's parameters to minimize a loss function, which measures the difference between the predicted classifications and the actual labels. In particular, we train our model applying an early stop on the best accuracy, adopting "AdamW" as optimizer, with a learning rate of 0.01, 50 epochs and testing two loss functions: Binary Cross-Entropy with Logits Loss and Cross Entropy Loss. Given the lack of prior research on the optimal loss function for this type of models, we assume that an empirical comparison was necessary.



(a) Requirement parsed with Bobcat Parser and parameterized with Tensor Ansatz.



(b) Requirement parsed with Cups Reader and parameterized with Tensor Ansatz.

Fig. 6. Examples of output of Tensor Ansatz from string diagrams.

Binary Cross-Entropy (BCE) treats classification as a set of independent binary decisions, allowing for a more flexible probability distribution over classes; Cross Entropy (CE) enforces a strict multiclass probability distribution, assuming that each instance belongs to only one class. Since tensor-based representations introduce new challenges in modeling classification probabilities, this comparative analysis helps determine whether a multi-label probabilistic approach (BCE) or a strict categorical formulation (CE) is more effective in this setting. After training, the model is evaluated on a separate test set to assess its ability to accurately classify unseen NFRs. The requirements sets are the same used in the experimentation for RQ1, detailed in Section 4.2.

4.3.3 ***Study design to answer RQ3.*** We follow the quantum path in Figure 1, performing two kind of quantum simulation: (a) a non-shot based experiment and (b) a noisy shot based experiment (see Figure 2), as detailed in Figure 7.

The Parsing and Encoding phase is the same adopted in the previous RQ2, ending with the representation of requirements as string diagrams (see Figure 5).

Only for (b) the noisy shot based experiment, we rewrite the diagrams to make it simpler for the training of the quantum model simulating scenario. In particular, we adopt the Unify Codomain Rewriter and the Remove Cups Rewriter. The former rewriter looks at the codomain of the diagram, and if this consists of more than one wires, it merges these wires with an extra box. The latter removes cups from a given diagram, because diagrams with less cups become circuits with less post-selection, which results in faster Quantum ML experiments. An example of the output of this phase is given in Figure 8.

The following step is to parameterize the diagrams into quantum circuits. This is done by applying a quantum circuit ansatz, which maps the abstract structure of the string diagrams to specific quantum gates and qubits. The ansatz

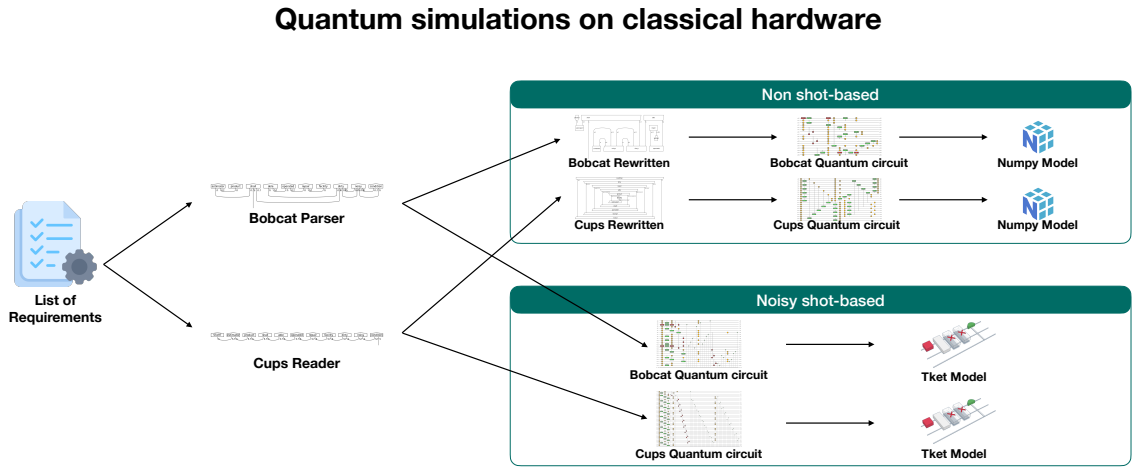
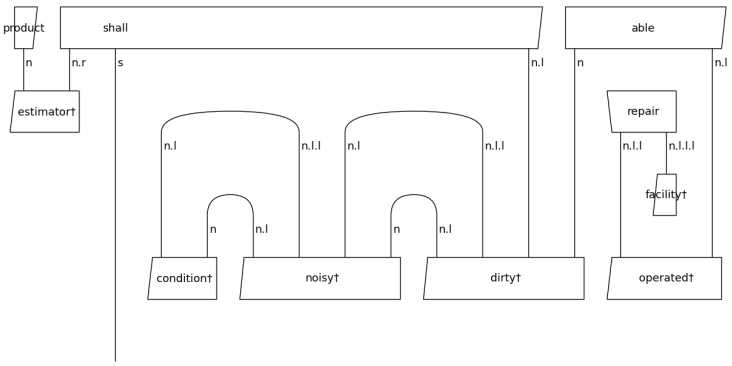


Fig. 7. Process applied to address RQ3.

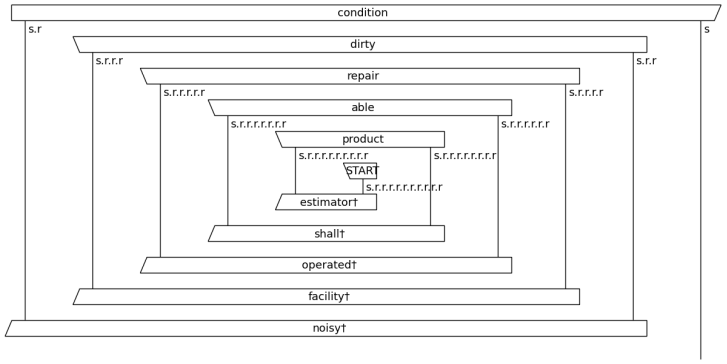
determines the number of qubits required and the type of quantum operations (gates) that will be applied to each component of the string diagram. In this study, we investigate the Instantaneous Quantum Polynomial (IQP) Ansatz, which interleaves layers of Hadamard gates with diagonal unitaries [16]. We choose IQP Ansatz because it is the standard approach in QNLP for converting string diagrams into quantum circuits, ensuring a direct and structured mapping from the textual representation to its quantum counterpart. It is particularly advantageous due to its (1) customizability, allowing for multiple adaptations and optimizations, making it possible to design lighter quantum circuits with reduced depth; (2) efficiency, by leveraging diagonal unitaries, it enables low-cost implementation while preserving essential linguistic dependencies; and (3) hardware feasibility, shallow circuits minimize quantum decoherence, making IQP well-suited for execution on current quantum devices. This approach ensures that our quantum circuits remain efficient, scalable, and adaptable, aligning with the goal of exploring QNLP representations while considering practical quantum simulations constraints.

Note that at this point we have in total four kind of diagrams to deal with. For the experiment (a), i.e. the non-shot based simulation, we have the diagrams output of the Bobcat parser and Cups Reader (Figure 5a and Figure 5b, respectively). For the experiment (b), i.e. the noisy shot based simulation, we have the diagrams output of the rewriting phase (Figure 8a and Figure 8b).

An important choice is made on the setup of the IQP Ansatz, which determines the structures of the created circuits. The setup of the IQP Ansatz is configured through a mapping of atomic types (such as sentences and nouns) to qubits, along with the definition of key parameters that control the structure and complexity of the quantum circuit. In our study, the IQP Ansatz is characterized by a single layer of quantum gates (n_layers=1), the use of multiple single-qubit parameters (n_single_qubit_params=3), and the option to discard additional qubits after processing (discard=True). Firstly, we define a mapping that assigns a specific number of qubits to each atomic type present in the string diagram. In our case “sentence” is mapped to 2 qubits, while “noun” is mapped to 1 qubit. This mapping determines how many qubits are allocated to represent the quantum states corresponding to sentences and nouns in the diagram. The choice of 2 qubits for sentences allows for a richer representation of their structure, while 1 qubit is used for nouns, reflecting their simpler structure in this context. The “n_layers” parameter is set to 1, indicating that the quantum circuit will



(a) Requirement parsed with Bobcat Parser and rewritten with Unify Codomain and Remove Cups rewriters.

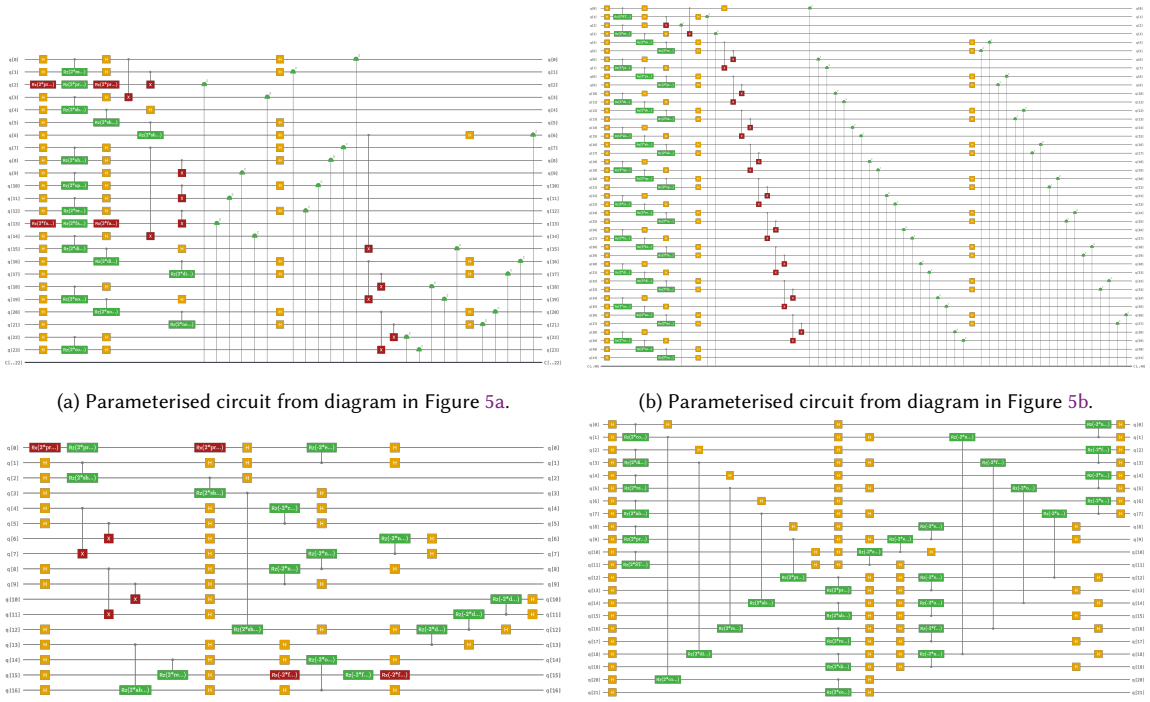


(b) Requirement parsed with Cups Reader and and rewritten with Unify Codomain and Remove Cups rewriters.

Fig. 8. Examples of output of Rewriting phase from string diagrams.

consist of a single layer of Hadamard gates interleaved with diagonal unitaries. This setup provides a basic level of entanglement and interaction between the qubits, suitable for capturing relationships in the data while keeping the circuit relatively simple. The “n_single_qubit_params” parameter is set to 3, meaning that each qubit will be associated with three parameterized single-qubit gates (such as rotations). These gates allow for fine-tuning the quantum states of individual qubits, enabling the circuit to capture more detailed information about the input data. This parameterization is crucial for adjusting the quantum circuit to effectively encode the information contained in the string diagrams. Lastly, the “discard=True” option indicates that any qubits that are not needed after the main computation will be discarded. Discarding excess qubits helps reduce the complexity of the circuit and avoids potential issues with post-selection, making the circuit more efficient and easier to simulate or execute on quantum hardware.

We use IQP Ansatz configured in this way for all of our experiments. Examples of resulting quantum circuits are given in Figure 9. A more detailed picture of each can be found in the shared notebooks in the replication package [8].



(a) Parameterised circuit from diagram in Figure 5a.

(b) Parameterised circuit from diagram in Figure 5b.

(c) Parameterised circuit from diagram in Figure 8a.

(d) Parameterised circuit from diagram in Figure 8b.

Fig. 9. Examples of output of IQP Ansatz from string diagrams.

The next step is the training and testing. For the (a) non-shot based experiment, we simulate a quantum environment using the Numpy Model, which leverages tensor networks simulations to represent the quantum circuits and their operations. We set "use_jit=True" parameter, enabling Just-In-Time (JIT) compilation, which optimizes the model's performance by speeding up the evaluation of the tensor networks. In this case, we only use the Cross Entropy Loss as the loss function, as it is the most suited one for the multiclass classification task and because we do not want to add another parameter to be analyzed at this stage, in contrast to the previous experimentation for RQ2. The model is trained for 50 epochs, while the SPSA Optimizer [46] is used for optimizing the model's parameters. Note that the optimizer's behavior is controlled by three hyperparameters: "a", the step size scaling factor or learning rate, set to 0.05; "c", the perturbation scaling factor, set to 0.06; and "A", a stability constant, set to 0.01 * 50 (number of epochs). This is the default configuration suited for most of the cases, and we use this one as our first intention is not to find the most effective setup, but rather to explore the various scenario provided in this field. The training process is carefully monitored, and early stopping on the accuracy metric is employed to prevent overfitting, ensuring the model's robustness and effectiveness in classifying non-functional requirements.

For the (b) noisy shot based experiment we use the AerBackend from the "pytket" library, which simulates the execution of quantum circuits on classical hardware. This backend is configured to mimic the behavior of a real quantum

device, including the execution of circuits in a shot-based manner. Such execution can be configured in different ways. In our study, the circuits are compiled using the default compilation pass with a level of 2, that ensures that the circuits are optimized for execution on the simulated quantum hardware, reducing errors and improving efficiency. The number of shots is set to 8192, meaning each circuit will be executed 8192 times. The use of multiple shots allows the model to estimate the probabilities of different outcomes, which is essential for tasks that rely on statistical results, such as classification. We employ the Tket Model that converts the quantum circuits into a format compatible with the AerBackend and prepares them for execution in a shot-based manner. The training of such model follow the same configuration of the experiment (a): Cross Entropy Loss as loss function, 50 epochs, SPSA as optimizer with the same parameters "a", "c" and "A" as in experiment (a). Again, we set the early stopping on the accuracy.

For both the experiment (a) and (b), we use the same train, test and validation set used in the previous RQs, so that we can make a comparison between the results obtained from the different use cases.

4.4 Evaluation Criteria

To evaluate the models we experiment with, we utilize well-established information retrieval metrics: Accuracy, Precision, Recall, and F1-score [3, 38].

Accuracy measures the proportion of correct predictions, including both *true positives* and *true negatives*, out of the total number of predictions made by the model (*true positives* + *true negatives* + *false positives* + *false negatives*). *Precision*, defined as *true positives* / (*true positives* + *false positives*), reflects the accuracy of the positive predictions made by the model. *Recall*, calculated as *true positives* / (*true positives* + *false negatives*), assesses the model's ability to identify all relevant instances. *F1-score* is the harmonic mean of Precision and Recall, providing a single metric that balances these two aspects. We focus on the weighted-average F1-score, which is computed by averaging the F1-scores of each class while accounting for the number of instances in each class.

We choose these criteria due to their importance in most if not all classification tasks. In our study, Recall should be particularly significant because stakeholders may prioritize a tool that identifies all potential NFRs, even if it results in some incorrect predictions [26]. Moreover, these metrics allow for comparisons between experiments we carry out and with existing or following works in literature, helping to assess and analyze the effectiveness or weaknesses of this approach. The objective is to evaluate the models' accuracy in identifying non functional requirements and to identify any limitations in this task. For evaluating the results, we consider a model to be effective if it achieves an F1-score greater than 0.7, a threshold that has been used in other studies as an indicator of model quality [18, 26].

4.5 Threats to validity

This section discusses the primary threats to the validity of our study, particularly focusing on the generalizability, repeatability, and accuracy of the methods and tools used.

Construct Validity. Construct validity pertains to how well the evaluation methods measure the concepts they are intended to assess, ensuring that the observations and inferences drawn from the study are appropriate [7]. In this work, we employed metrics from the Scikit-learn library, including the `accuracy_score`[5] for Accuracy, `precision_score`[6] for Precision, `recall_score`[7] for Recall, and `f1_score`[8] for F1-score.

[5] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html
[6] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html
[7] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html
[8] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

Although relying on a single tool could introduce some concerns, especially in ML contexts, these metrics are based on mathematical formulas, making them independent of the specific tool used. Additionally, these metrics were chosen because they are widely used in prior studies on NFR detection, enabling meaningful comparisons with previous and further researches.

Note that for ML and word embedding approaches, we apply cross-validation to enhance model reliability. However, for QNLP, we adopt a hold-out validation strategy due to the computational cost of quantum simulations, particularly in shot-based noisy settings, where cross-validation would significantly increase execution time. Additionally, comparing different requirement representations requires consistent evaluation across all approaches, which would necessitate manual fold selection to ensure each method is tested on identical subsets.

Internal Validity. Internal validity is concerned with the degree to which the results can be attributed to the interventions applied, rather than other factors [7]. In our study, we assume comparability among the models because they are derived from the same libraries, such as Scikit-learn in RQ1, and lambeq in RQ2 and RQ3. Since causality could pose a threat to internal validity, we are aware that the results are the product of chance, mainly because the choices made in the course of the experiments are varied, starting from the preprocessing of the dataset to the selection of models to be adopted in the different scenarios. Further experiments are needed, and to support that we design a possible road map in Section 6 to help and stimulate the research in the field.

External Validity. External validity relates to the generalizability and reproducibility of the study's findings [7], which determines the applicability of our results in real-world scenarios.

One limitation of our study is the use of a subset of the PROMISE expanded dataset, focusing primarily on shorter requirements. This decision was necessary due to the computational challenges associated with converting long requirements into string diagrams, tensor networks, and quantum circuits. Unlike classical NLP approaches, which can handle long sequences efficiently, QNLP techniques require encoding text into quantum states, leading to an exponential increase in circuit complexity as sequence length grows. This limitation may impact the generalizability of our findings, as longer requirements could introduce additional challenges not captured in our experiments. Future work should explore methods to process longer requirements, such as hybrid quantum-classical techniques, dimensionality reduction in tensor networks, or optimized quantum encodings.

Our approach utilizes Python-based models, specifically those implemented using available Python libraries. The methods employed in this study may not be directly transferable to other programming languages, and since we do not apply validation techniques focused on generalizability like cross validation, the external validity may be dependent on the specific training data. However, the datasets used span over various projects within software specifications, supporting the applicability of the approach in different industrial settings. To facilitate the replication of our study, all tools, scripts, and data have been made publicly available [8].

For our experiments, we utilized the NVIDIA L4 GPU available on Google Colab, which is based on the Ada Lovelace architecture. The L4 GPU is optimized for deep learning tasks, offering 24 GB of memory and 4th-generation Tensor Cores. This setup provided the necessary computational power to efficiently train and evaluate our models across different ML- and quantum-based approaches.

Conclusion Validity. Conclusion validity refers to the soundness of the inferences drawn from the data and the reliability of the conclusions based on the employed metrics [7].

We recognize that the conclusions regarding the effectiveness of the ML models in our RQs are based on metrics like Precision, Recall and so on. We acknowledge that these metrics alone may not capture the full performance potential of

the models. Additional experimentation and statistical tests could provide a deeper insight into the model's efficacy, thereby refining the conclusions presented in this study.

5 Analysis of the Results

In this section, we present the results of our investigation for each research questions formulated in Section 4.1.

5.1 RQ1. How effectively can shallow machine learning algorithms classify non-functional requirements when represented with word embeddings techniques?

We summarize the results in Table 3, which provides a comparison of selected word embedding techniques (TF-IDF, BERT, fastText, Word2vec, and GloVe) and their performance when combined with different ML models. Each technique is evaluated based on the accuracy and F1-score of the models, and a detailed analysis is provided by examining the best model performance on a hold-out test set.

For TF-IDF, performance remains relatively low across all models (Table 3a). The Random Forest Classifier performs best, achieving an accuracy of 0.39 and an F1-score of 0.36. However, on the hold-out set (Table 3b), results reveal poor precision and recall, especially for the Security (SE) class, which fails to produce any correct classifications. BERT embeddings show a notable improvement over TF-IDF (Table 3c). The Ridge Classifier CV delivers the best performance with an accuracy of 0.55 and an F1-score of 0.54. Despite these gains, hold-out set results (Table 3d) remain moderate, with Operability (O) and Performance (PE) classes achieving the highest scores. However, challenges persist with the Security (SE) and Usability (US) classes, which record lower F1-scores. Moving to fastText embeddings, we observe superior performance compared to both TF-IDF and BERT. The SVC (Support Vector Classifier) model achieves an accuracy of 0.66 and an F1-score of 0.66. On the hold-out set (Table 3f), results remain consistent across all classes, with an average F1-score of 0.68. The Security (SE) class demonstrates significant improvement, reaching an F1-score of 0.70, which is higher than what is observed with previous embeddings. Word2vec provides even stronger results, with the Random Forest Classifier achieving an accuracy of 0.77 and an F1-score of 0.78. The hold-out set (Table 3h) further confirms this, particularly with the Performance (PE) class, which reaches an impressive F1-score of 0.92. The average F1-score across all classes stands at 0.81, highlighting the effectiveness of Word2vec for NFR classification tasks. GloVe embeddings outperform all other techniques, with the NuSVC model delivering the highest accuracy of 0.82 and an F1-score of 0.82. On the hold-out set (Table 3j), results are outstanding, with an average F1-score of 0.84. Precision and recall remain balanced across all classes, and the Security (SE) class achieves an F1-score of 0.80, making GloVe the most effective embedding technique for the NFRs classification task.

Summary of the Results. The best overall performance is achieved using GloVe embeddings combined with the NuSVC model, which demonstrates superior results across all metrics. Word2vec also provides interesting results when combined with Random Forest Classifier, slightly worse than GloVe. While fastText achieves a balanced and acceptable performance across classes when combined with the SVC model, BERT fails to achieve 0.50 in F1-Score in 3 out of 4 classes. TF-IDF, while expected to be effective for simpler tasks, is less suitable for the task of NFR classification.

Table 3. Comparison of selected word embedding techniques with ML models. On the left side, the selection of the best model based on a particular embedder. On the right, the best model tested on unseen data (hold out).

(a) Best models based on TF-IDF embedding.

Model	Accuracy	F1-score
Random Forest Classifier	0.39	0.36
Nearest Centroid	0.36	0.36
Gaussian NB	0.34	0.33

(b) Test on hold out set of Random Forest Classifier based on TF-IDF embedding.

Class	Precision	Recall	F1-score
O	0.23	0.55	0.32
PE	0.12	0.09	0.11
SE	0.00	0.00	0.00
US	0.20	0.09	0.13
Average	0.14	0.18	0.14

(c) Best models based on BERT embedding.

Model	Accuracy	F1-score
Ridge Classifier CV	0.55	0.54
NuSVC	0.52	0.53
Ridge Classifier	0.50	0.49

(d) Test on hold out set of Ridge Classifier CV based on BERT embedding.

Class	Precision	Recall	F1-score
O	0.41	0.64	0.50
PE	0.57	0.36	0.44
SE	0.31	0.36	0.33
US	0.29	0.18	0.22
Average	0.39	0.39	0.38

(e) Best models based on fastText embedding.

Model	Accuracy	F1-score
SVC	0.66	0.66
Gaussian NB	0.61	0.61
Extra Tree Classifier	0.59	0.59

(f) Test on hold out set of SVC based on fastText embedding.

Class	Precision	Recall	F1-score
O	0.67	0.73	0.70
PE	0.73	0.73	0.73
SE	0.78	0.64	0.70
US	0.58	0.64	0.61
Average	0.69	0.68	0.68

(g) Best models based on Word2vec embedding.

Model	Accuracy	F1-score
Random Forest Classifier	0.77	0.78
Logistic Regression	0.77	0.77
Passive Aggressive Classifier	0.77	0.77

(h) Test on hold out set of Random Forest Classifier based on Word2vec embedding.

Class	Precision	Recall	F1-score
O	0.75	0.82	0.78
PE	**0.85**	**1.00**	**0.92**
SE	0.78	0.64	0.70
US	0.90	0.82	0.86
Average	0.82	0.82	0.81

(i) Best models based on GloVe embedding.

Model	Accuracy	F1-score
NuSVC	0.82	0.82
SVC	0.80	0.80
Random Forest Classifier	0.80	0.79

(j) Test on hold out set of NuSVC based on GloVe embedding.

Class	Precision	Recall	F1-score
O	**0.82**	**0.82**	**0.82**
PE	0.79	1.00	0.88
SE	**0.89**	**0.73**	**0.80**
US	**0.90**	**0.82**	**0.86**
Average	**0.85**	**0.84**	**0.84**

5.2 RQ2. How effectively can a basic model classify non-functional requirements when represented as string diagrams and parameterised as tensor networks?

To answer RQ2 we analyzed the performance of classical Pytorch models using Bobcat-based and Cups-based tensor networks for classifying NFRs. The training process involved two separate setups, one using Binary Cross-Entropy (BCE) loss and the other using Cross Entropy (CE) loss.

Firstly, let's analyze the Bobcat-based experiment. The model's training and validation performance, shown in Figure 10a over the course of 43 epochs using BCE loss show substantial improvement in training metrics while validation metrics stabilize. During Epoch 1, the training accuracy was 0.23, with an F1-score of 0.52, and by Epoch 10, training accuracy had increased to 0.97, and the F1-score had reached 0.96. However, the validation accuracy remains at around 0.27, and the validation F1-scores hovered between 0.25 and 0.36 during the entire training period. The early stopping criterion is triggered at Epoch 43, selecting the best model at epoch 20 and suggesting that further training would likely result in overfitting. After training, we test the model on a hold-out dataset, obtaining the results shown in Table 4a. A class-wise performance revealed that class (O) achieved the best precision and recall, with an F1-score of 0.35. Other classes showed much lower precision, recall, and F1-scores, particularly usability class, where the model struggled to achieve meaningful predictions.

Using Cross Entropy (CE) loss, the training results (Figure 10b) show a similar trend to the BCE training setup, with better training metrics but a slower convergence for validation metrics. During Epoch 1, the training accuracy started at 0.23, and by Epoch 10, it increased to 0.98. The validation accuracy stabilize around 0.36, with an F1-score in the 0.34 to 0.36 range by Epoch 10. Early stopping was triggered at Epoch 24, with the model reaching its peak performance at Epoch 8. We test the best model on the hold-out set, producing the results shown in Table 4b. Although the overall accuracy is low, the CE-based model shows a slight improvement in class 0 predictions, but the other classes, particularly usability class, still shows weak precision and recall.

Table 4. Comparison of Pytorch models trained on Bobcat-based tensor networks.

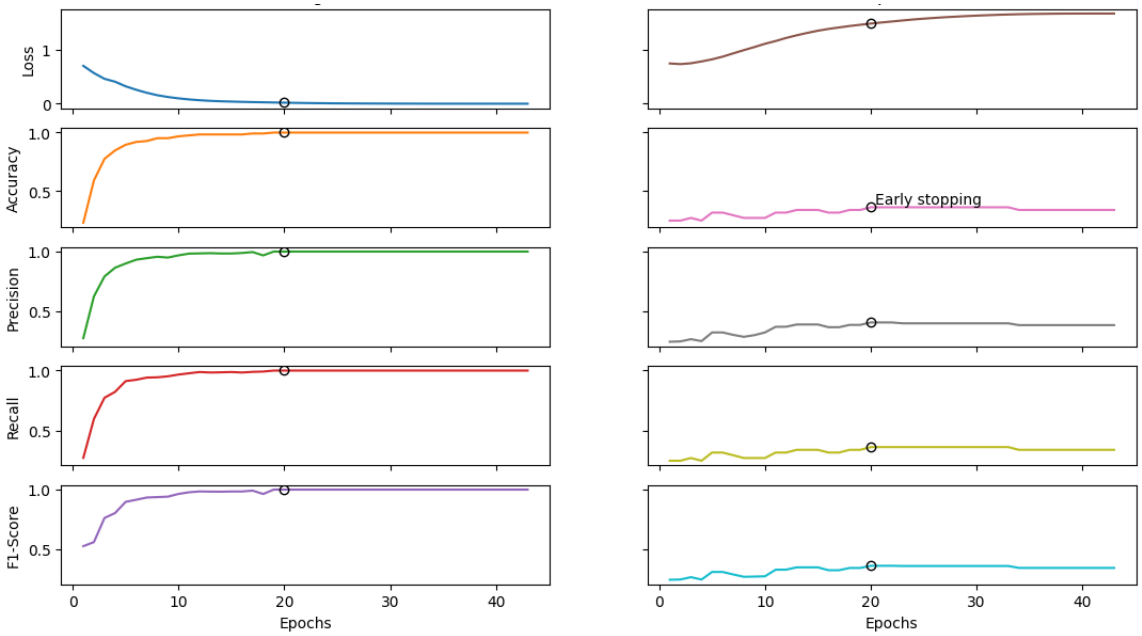
(a) Results using Binary Cross Entropy with Logits Loss during training and test on hold out set.

Class	Precision	Recall	F1-score
O	0.31	0.40	0.35
PE	0.22	0.18	0.20
SE	0.20	0.18	0.19
US	0.09	0.09	0.09
Average	0.21	0.21	0.21

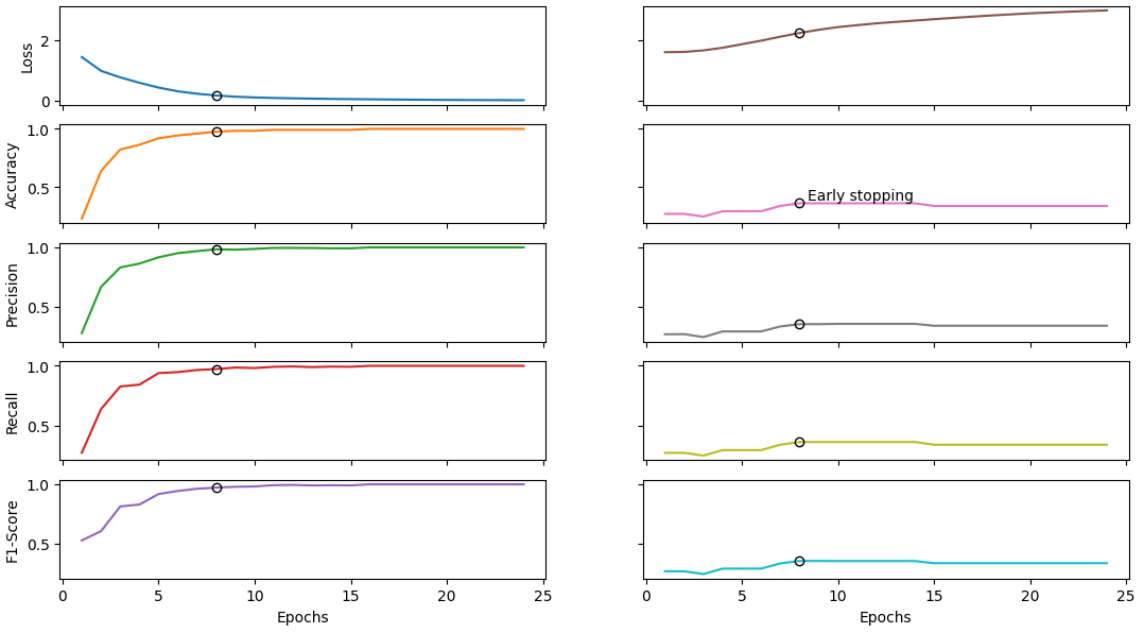
(b) Results using Cross Entropy Loss during training and test on hold out set.

Class	Precision	Recall	F1-score
O	0.40	0.60	0.48
PE	0.11	0.09	0.10
SE	0.18	0.18	0.18
US	0.00	0.00	0.00
Average	0.17	0.22	0.19

During training on Cups-based tensor networks with the BCE loss, the model shows notable improvements across the first few epochs but begins to overfit as validation metrics stabilize, as shown in Figure 11a. At Epoch 1, results are modest, with a training Accuracy of 0.23 and an F1-score of 0.56. The validation accuracy is 0.25, with a lower F1-score of 0.24. At Epoch 22 the best model is identified, with training accuracy peaking at 0.98 and an F1-score of 0.98. Validation accuracy is 0.36, with an F1-score of 0.36, signaling limited generalization capacity. Early stopping is triggered, and this model is saved for evaluation on the hold-out set. This model is evaluated on the hold-out set, yielding the results displayed in Table 5a. Again, the model achieves the best performance in operability class, while on other classes it shows relatively lower results, by reaching an average F1-score of 0.23, indicating moderate performance across the classes.

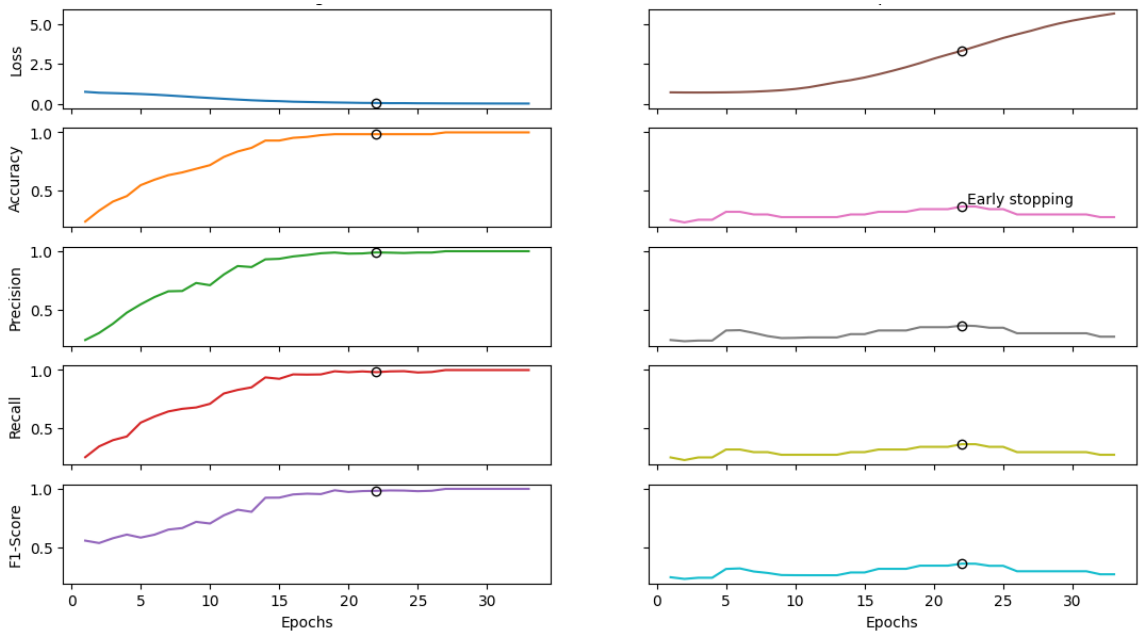


(a) Performance in training with BCE on training set (left) and validation set (right).

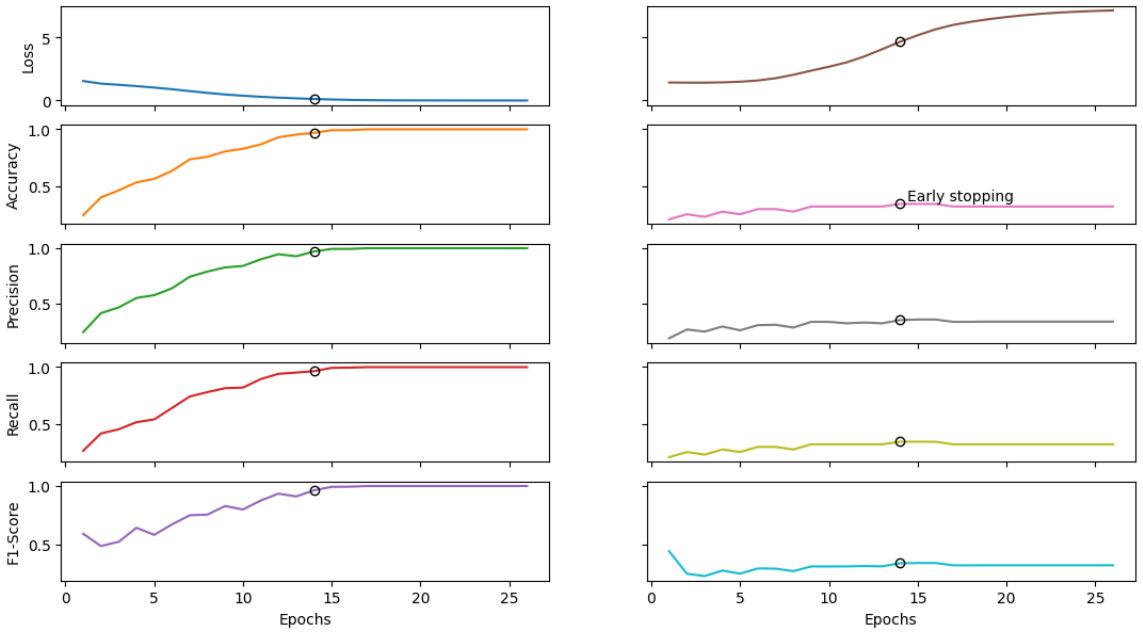


(b) Performance in training with CE on training set (left) and validation set (right).

Fig. 10. Performance during training with Bobcat-based tensor networks.



(a) Performance in training with BCE on training set (left) and validation set (right).



(b) Performance in training with CE on training set (left) and validation set (right).

Fig. 11. Performance during training with Cups-based tensor networks.

For the experiments using CE loss, the Pytorch model demonstrates modest initial training performance but faces challenges in achieving generalization, as seen in the validation performance in Figure 11b. At Epoch 1 training accuracy is 0.24 with an F1-score of 0.59, while the validation accuracy is 0.20 and the F1-score is 0.44. At Epoch 14 the model is saved as the best-performing model, with 0.97 training accuracy and an F1-score of 0.96. Validation accuracy is 0.34, and the F1-score is 0.34. Early stopping is triggered at this point. The best CE model is evaluated on the hold-out set, providing the results shown in Table 5b. Also in this case, on the operability class, the model has the highest F1-score at 0.56, followed by security class with 0.50, but showing lower performance in usability and performance classes, with F1-scores of 0.27 and 0.33, respectively. The average F1-scores is 0.42, indicating a reasonable but still moderate performance across the classes.

Table 5. Comparison of Pytorch models trained on Cups-based tensor networks.

(a) Results using Binary Cross Entropy with Logits Loss during training and test on hold out set.

Class	Precision	Recall	F1-score
O	0.30	0.27	0.29
PE	0.18	0.27	0.21
SE	0.25	0.18	0.21
US	0.22	0.18	0.20
Average	0.24	0.23	0.23

(b) Results using Cross Entropy Loss during training and test on hold out set.

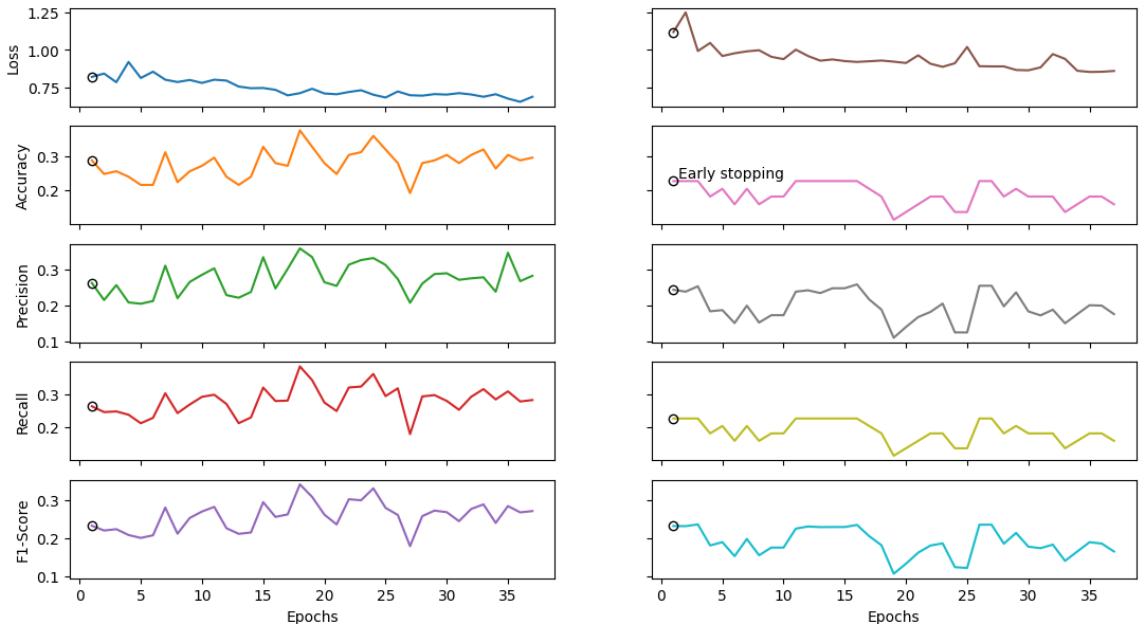
Class	Precision	Recall	F1-score
O	**0.71**	**0.45**	**0.56**
PE	**0.31**	**0.36**	**0.33**
SE	**0.46**	**0.55**	**0.50**
US	**0.27**	**0.27**	**0.27**
Average	**0.44**	**0.41**	**0.42**

Summary of the Results. The experiments using Bocat-based and Cups-based tensor networks for NFRs classification show moderate effectiveness. The best results are achieved in Cups-based experiment with Cross Entropy Loss, reaching a precision of 0.44, a recall of 0.41 and F1-score of 0.42. The best-performing class across all models is Operability. Overall, while tensor networks demonstrate promise, further optimization is needed to improve generalization and performance across all NFR classes.

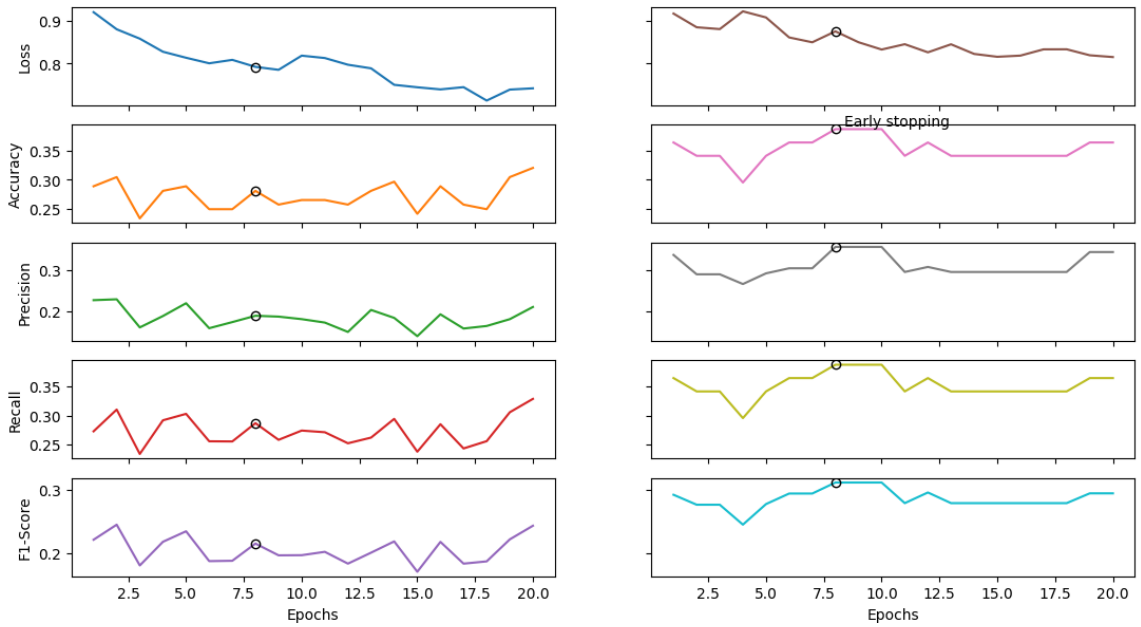
5.3 RQ3. How effectively can quantum models classify non-functional requirements when represented as string diagrams and parameterised as quantum circuits?

As anticipated in Section 4.1, to answer RQ3 we perform two experiments: (a) non-shot based experiment and (b) a noisy shot based experiment (see Figure 2). In the experiment (a), the results obtained during training are shown in Figure 12.

In particular, the results of training a Numpy model based on quantum circuits generated from the Bobcat parser show moderate performance with consistent challenges across the epochs (Figure 12a). In the initial epoch, the model starts achieving a training accuracy of 0.29 and a validation accuracy of 0.23. The corresponding F1-scores of 0.23 for training and 0.23 for validation indicate an underfitting trend. Over the next few epochs, the model's performance fluctuates slightly, with training accuracy peaking at around 0.31 in epoch 7, while validation accuracy remains stagnant at 0.23. The precision and recall metrics similarly reflect the model's struggle to generalize. For instance, the precision for validation oscillates around the 0.24-0.26 range across most epochs, while recall stays relatively stable at 0.23, with minor improvements. The model continues to suffer from low F1-scores, indicating it struggles with both overfitting and generalization issues throughout training.



(a) Performance with Bobcat-based quantum circuits on training set (left) and validation set (right).



(b) Performance with Cups-based quantum circuits on training set (left) and validation set (right).

Fig. 12. Performance during training with Numpy model on quantum circuits.

The best model, selected after the first epoch, is applied to the hold-out set, with the results shown in Table 6a. The evaluation results indicate a hold-out accuracy of 0.30, which is marginally better than the validation accuracy but still quite low. The precision (0.33) and recall (0.30) suggest that while the model can make some correct predictions, it perform consistently across all classes. Indeed, the F1-scores between classes fluctuate in range 0.24-0.35, indicating that some features help the model to distinguish between classes.

The training results from the Cups-based quantum circuits experiment (Figure 12b) exhibit fluctuating performance across the epochs, with gradual improvement in validation metrics but still limited classification effectiveness. Epoch 1 starts with a training accuracy of 0.29 and validation accuracy of 0.36 reflect the model's initial struggle to classify the non-functional requirements effectively. The F1-scores are 0.22 (training) and 0.29 (validation), indicating poor generalization. Over next epochs, the training precision and recall fluctuate, showing modest increases, especially by Epoch 8, where validation accuracy reaches 0.39, with an F1-score of 0.31. However, the training precision remains low throughout, and the model shows difficulties in learning patterns effectively from the dataset. Once the best model (saved at Epoch 8) is applied to the hold-out set, the overall performance remains modest (Table 6b). Operability class performs the best in terms of both recall and F1-score, though precision remains moderate. In Performance class, the model struggles with detecting this, resulting in a low recall and F1-score. While recall is higher in Security class than Performance class, the precision is still below acceptable thresholds. However, the model is unable to predict the Usability class at all.

Table 6. Comparison of Numpy models trained on quantum circuits.

(a) Results using Numpy models with train on Bobcat-based quantum circuits and test on hold out set.

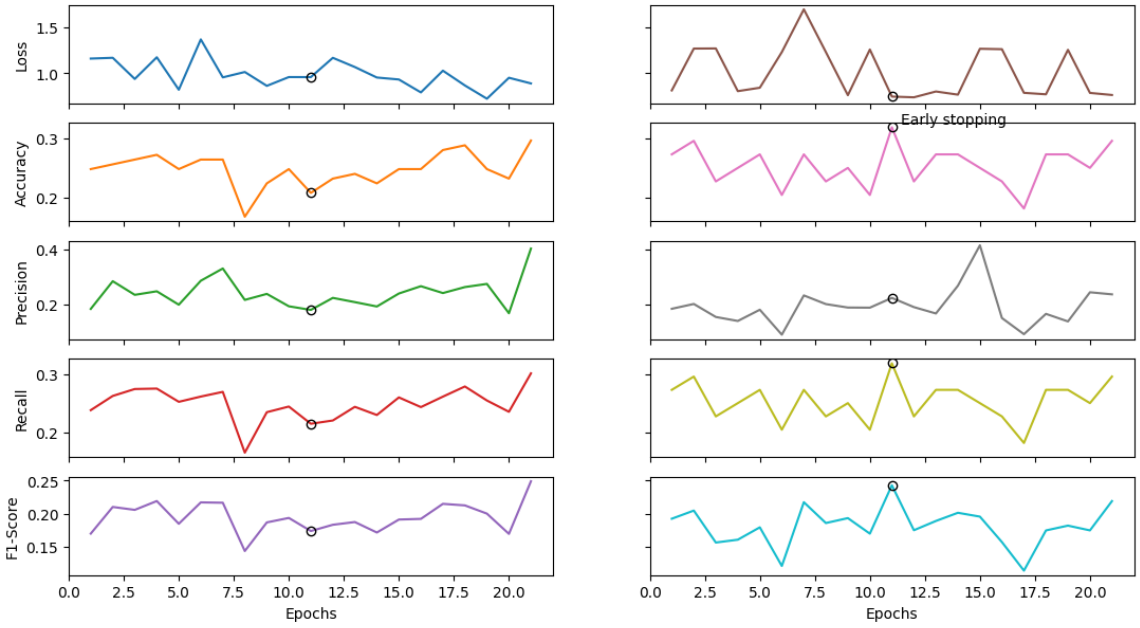
Class	Precision	Recall	F1-score
O	0.22	0.40	0.29
PE	**0.43**	**0.27**	**0.33**
SE	0.33	0.18	0.24
US	**0.33**	**0.36**	**0.35**
Average	**0.33**	**0.30**	**0.30**

(b) Results using using Numpy models with train on Cups-based quantum circuits and test on hold out set.

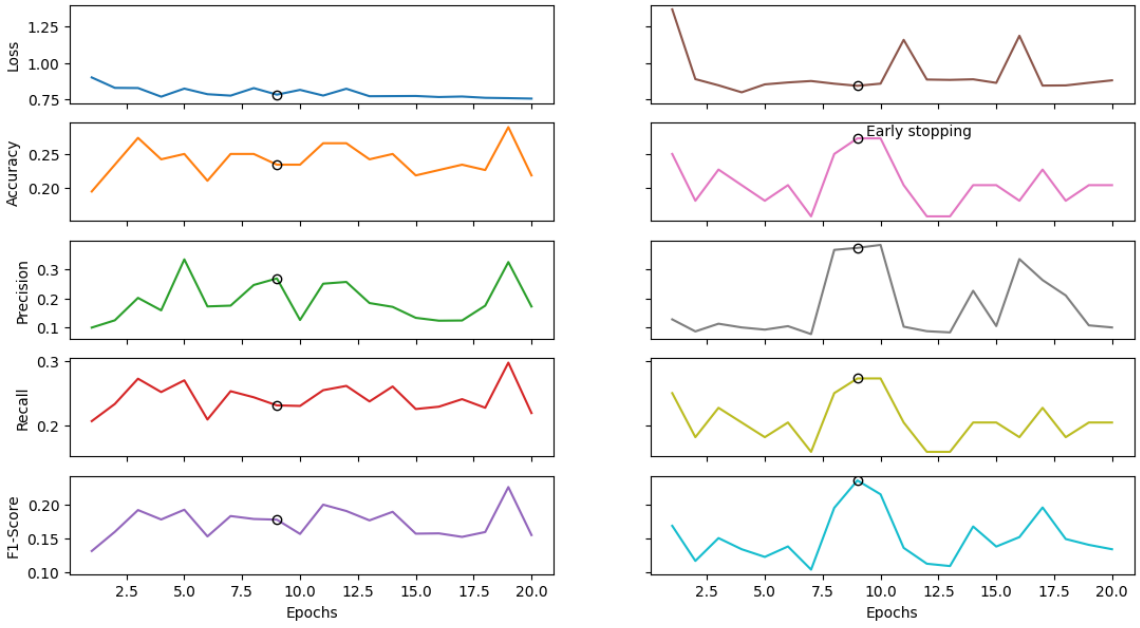
Class	Precision	Recall	F1-score
O	0.30	0.55	0.39
PE	0.25	0.09	0.13
SE	0.25	0.45	0.32
US	0.00	0.00	0.00
Average	0.20	0.27	0.21

In the (b) shot-based experiment, we train the Tket quantum model using quantum circuits created from both the Bobcat parser and Cups Reader and executed on the AER Backend. The results of Tket training on Bobcat-based quantum circuits demonstrate variability in model performance across different epochs, with generally low precision, recall, and F1-scores, indicating limited learning from the data (Figure 13a). Epoch 1 starts with a low training accuracy of 0.25 and validation accuracy of 0.27. The initial F1-scores are 0.17 (training) and 0.19 (validation), showing that the model is not learning effectively. As the epochs progress, the training precision and recall fluctuate without significant improvement. Epoch 11 shows the best performance with training accuracy of 0.30 and validation accuracy of 0.32, alongside an F1-score of 0.24 for the validation set. The model stabilizes by this epoch, but performance remains limited. After applying the best model (from Epoch 11) to the hold-out set, the overall performance remains weak, as can be seen in Table 7a. The model performs best on Operability class, with relatively high recall but low precision. On the Performance class the results are moderate, with low recall limiting the model's effectiveness. The remaining classes both exhibit 0.00 on precision, recall, and F1-scores, indicating the model's complete inability to classify these classes.

The same experiment conducted using quantum circuits created from the Cups Reader also demonstrate challenges in learning, with low precision, recall, and F1-scores across most epochs, indicating that the model struggles to classify



(a) Performance with Bobcat-based quantum circuits on training set (left) and validation set (right).



(b) Performance with Cups-based quantum circuits on training set (left) and validation set (right).

Fig. 13. Performance during training with Tket model on quantum circuits.

non-functional requirements effectively (Figure 13b). Epoch 1 starts with poor performance in both precision and recall. Training accuracy is 0.20, while validation accuracy is 0.25, showing that the model is struggling to generalize from the data. The F1-score for training is 0.13 and 0.17 for validation. As training continues, precision and recall remain inconsistent. The best training performance is seen in Epoch 9, with training precision reaching 0.27 and F1-scores of 0.24 on the validation set, reflecting moderate improvement in some classes. Early stopping is triggered after Epoch 20, as the model fails to make significant gains in performance, and the best model from Epoch 9 is saved. After applying the best model from Epoch 9 to the hold-out set, the performance varies between classes, with the Operability class achieving 0.32 precision and 0.64 recall, but Performance and Usability class exhibit 0.00 precision, recall, and F1-scores, showing that the model fails to detect certain classes.

Table 7. Comparison of Tket models trained on quantum circuits.

(a) Results using Tket model with train on Bobcat-based quantum circuits and test on hold out set.

Class	Precision	Recall	F1-score
O	0.23	0.70	0.34
PE	0.40	0.18	0.25
SE	0.00	0.00	0.00
US	0.00	0.00	0.00
Average	0.16	0.22	0.15

(b) Results using Tket model with train on Cups-based quantum circuits and test on hold out set.

Class	Precision	Recall	F1-score
O	**0.32**	**0.64**	**0.42**
PE	0.00	0.00	0.00
SE	**0.25**	**0.45**	**0.32**
US	0.00	0.00	0.00
Average	0.14	0.27	0.19

Summary of the Results. Across all four experiments, the quantum models based on both Bobcat Parser and Cups Reader demonstrate limited effectiveness in classifying non-functional requirements. Both the shot-based and non-shot-based approaches encountered significant challenges in learning from the data, with low accuracy and F1-scores across the results. The best overall performance is observed in the non-shot-based Bobcat experiment, where the model achieved 0.30 in F1-score. However, this performance demonstrate limited learning capacity, and further optimization of quantum algorithms and models is needed before quantum circuits can be effectively applied to this domain.

6 Discussion and Research Roadmap

In this study, we explore three distinct computational approaches to classifying NFRs: shallow ML algorithms using word embeddings, tensor network-based models, and quantum circuit-based models. Each approach provides unique insights and raises important considerations regarding their strengths and limitations in NFR classification tasks. In the following, we discuss the findings coming from the analyses performed to answer the formulated research questions and compare the efficacy of the employed approaches in light of their respective methodologies and results.

6.1 Lesson learned

The results of RQ1 inform us that the combination of shallow ML algorithms with word embeddings result in very high performing ML models, classifying unseen non-functional requirements during the training phase with precision, recall, and f1-score metrics exceeding 0.80. In particular, GloVe and Word2Vec are predominant, with the former performing slightly better, although statistical tests should be conducted to confirm the significant difference. However, our focus is not to understand which combination is the most effective for the task at hand, but to understand where the new

solutions offered by both textual representation in string diagrams and quantum computation rank. Thus, while on the one hand we have combinations of ML algorithms with word embedding that reach F1-score over 0.80, on the other hand we have combinations that do not reach 0.40 of F1-score in an attempt to classify non-functional requirements that do not belong to the training sets, as in the case of TF-IDF (where performance is below 0.20) and BERT (where performance is around 0.38). These results suggest that word embeddings techniques based on word frequency (TF-IDF) and techniques that generate contextual embeddings (BERT) have difficulties in the classification task as we set it up.

As for the comparison with results of previous investigations, the key difference between our RQ1 findings and the results summarized in Table 1 lies in the fact that each study utilized a different dataset, targeted specific NFRs, applied distinct preprocessing techniques, employed various word embedding methods, and adopted different validation strategies. Since our work introduces a novel approach to representing requirements for training ML and quantum models, we deemed it crucial to conduct an initial exploration that enables a fair comparison—using a consistent dataset, identical metrics, and the same validation method. While our study confirms that shallow ML models combined with statistical word embeddings (e.g., GloVe, Word2Vec) consistently achieve F1-scores above 0.80, the results also indicate that BERT embedding struggles in scenarios where NFRs are not seen during training, with performance around 0.38, and TF-IDF performing even worse (below 0.20). In contrast, Table 1 presents BERT-based models (e.g., BERT-CNN, NFRNet) achieving F1-scores as high as 0.91–0.94, suggesting that fine-tuning on domain-specific datasets significantly enhances BERT's effectiveness. This reinforces that while BERT may not generalize well to unseen NFRs, it can perform exceptionally well when trained on specialized datasets. Additionally, the table highlights ensemble deep learning models (CNN, LSTM, GRU) as top performers, whereas our RQ1 results focused on comparing word embeddings rather than the effectiveness of deep architectures.

> 👉 In summary, RQ1 results differentiate statistical embeddings, such as GloVE, Word2vec, FastText, which consider words independently of context, from embeddings based on word frequency, such as TF-IDF, and from contextual embeddings, such as BERT. In particular, statistical embeddings achieve more than satisfactory results, while the rest have difficulties, due to both the volume of the data under consideration and the preprocessing steps used during the study.

As a first exploration in representing requirements in string diagrams, we conducted experiments exploiting Bobcat Parser and Cups Reader. This choice is due to the fact that the former is a syntax-based model and the latter is a word sequence-based model. Specifically, Bobcat Parser is a traditional symbolic model focused on syntax and grammar, used in rule-based NL tasks. Cups Reader is a model inspired on Quantum NLP since it aims at the representation of language as a network of tensors that can be easily manipulated in quantum computational frameworks.

Experimentation results exploited to answer RQ2 suggest that the representation via Cups Reader performs better than Bobcat Parser, especially using Cross Entropy as loss function in PyTorch model training (Table 5b). Note that for all our experiments we used the default version of the model, which simply exploits the tensors contraction to provide the output. This implies several considerations. First, we can evaluate the results obtained from simply contracting the tensors obtained from Bobcat Parser and Cups Reader, allowing a "direct" comparison between the different representations in string diagrams of the requirements. Next, since the basic model is a simple contraction operation, we can consider this model not very different from shallow ML models that are still based on more or less complex mathematical functions, allowing us to compare with the results obtained in response to RQ1. Finally, since the model is based on PyTorch, it is possible to build from it more complex neural networks that could positively or negatively

affect the final performance, in addition to the possibility to load the weights and symbols from a training checkpoint, allowing for an incremental learning implementation.

As a matter of fact, in the experimentation to answer RQ1 we evaluated the efficiency of different ML algorithms, while for RQ2, exploiting the basic model, we did not obtain astonishing performance. However, the results obtained from Cups Reader (Table 5b) are better in 3 classes (Operability, Security, and Usability) out of 4 than the combination of BERT and shallow ML (Table 3d), which presents better results only in identifying the Performance class.

> 👉 Although the results may be due to the case at hand, as with other requirements or with other configurations for tensor networks the result may have been different, the representation in string diagrams is comparable with contextual embeddings techniques and performs better than techniques that consider words frequencies.

The representation of requirements in string diagrams opens the way for comparison with the results obtained in the experiment carried out to answer RQ3, in which we represent string diagrams in quantum circuits. Clearly, here the factors influencing the requirements classification are even more numerous than in the previous case. In fact, the parameterization of the tensor diagrams in the experimentation performed to answer RQ2 was done through the definition of a sentence and noun mapping within the requirements by assigning dimension 4 to each, since 4 are the possible classes to which each belongs. However, this is also a questionable design choice, as we could have defined larger or smaller dimensions and modified the PyTorch model accordingly to shape the output of the neural network in 4 possible classes.

In parameterizing the diagrams in quantum circuits the situation becomes even more "complicated". In addition to defining the number of qubits to be assigned to each sentence and noun, one must also define the number of Hadamard gates layers (n_layer parameter) and the number of rotations assigned to each qubit. The choice of these parameters influences the transformation of string diagrams into quantum circuits, which may consequently be more or less suitable for the problem under consideration.

Considering these factors, additional choices are made in the simulation of quantum environments on classical hardware. First of all, the Numpy model we use in the non-shot based simulation converts quantum circuits into tensor networks, which on the one hand allows a comparison with the experimentation done in RQ2 since both are based on tensor network, and on the other hand is a bit counter-intuitive if one thinks of wanting to exploit the potential of quantum computation for the task of requirements classification.

As for the shot-based simulation, we strongly believe that the configuration of the backend used is another very influencing factor. Indeed, out of the choice of which backend to use, we have to define the compilation pass and the number of shots. The compilation pass refers to the process of transforming, optimizing, and preparing a quantum circuit to be executed on a specific backend. Shots refer to the number of times a quantum circuit is executed or simulated. In quantum computing, the results of measuring a qubit are probabilistic rather than deterministic, meaning each run of the circuit can yield different measurement outcomes. All these considerations indicate that the results we obtained in our experimentation may not only be the result of chance, but may also be a consequence of an ideal setup for the task under consideration. In addition, while in the setup of experiments based on ML and word embeddings there may be random components that influence the results obtained, in the quantum case the probabilistic components multiply, making it almost impossible to think of testing them all and understanding which ones influence the results more than others.

Analyzing the results for RQ3, obtained using the same setup adopted for RQ1 and RQ2 (i.e. same dataset, same task, same metrics, and same hold out validation), we can conclude that in both types of quantum environment simulation the results leave something to be desired. However, despite the probabilistic nature of these simulations, in some cases the results are shown to be comparable to those obtained in the other experiments. In detail, only in the non-shot based simulation we have obtained a model capable of recognizing all types of NFRs by exploiting Bobcat-based representation (Table 6a), albeit with relatively poor performance. In the remaining cases, there are classes that fail to be identified. In the shot-based simulation, the Operability class exhibits high recall in both cases, even better than the non-shot based simulation, but the remaining results on the other classes are really scarce.

When comparing the best results obtained for RQ2 (Table 5b) with the best results for RQ3 (Table 6a), the latter perform relatively worse, with a deviation of 0.10 on F1-score. However, although they are not the best, they are still better than the results obtained using TF-IDF and ML techniques in combination.

👉 In the non-shot based simulation, Numpy model converts quantum circuits into tensor networks, aligning it with RQ2's approach, which also uses tensor networks. However, this may seem counter-intuitive given that quantum computation's potential is not fully leveraged. In shot-based simulations, circuits configuration, compilation passes, and the number of shots play critical roles in influencing outcomes. While the results for RQ3 do not outperform those of RQ2, they show comparability, especially in identifying specific NFR classes. However, overall performance is lower than the tensor-based models in RQ2, with a 0.10 F1-score deviation, though still better than the combination of TF-IDF and ML methods.

6.2 Future Research Directions in NFR classification

Based on the methodology we adopted, in the following we want to share how researchers can organize the future directions for research in quantum-based non-functional requirements (NFR) classification, which is graphically summarized in Figure 14.

Firstly, future work could expand on task complexity by experimenting with binary or multi-label classification rather than the multi-class classification approach used in this work. This would enable a more nuanced understanding of the interdependencies between NFR categories, which often overlap in real-world applications.

An important direction for future research is expanding the dataset both in size and diversity. Including more instances from various software projects or domains would enhance the robustness of the models, potentially leading to better generalization. Additionally, expanding to cross-project datasets would provide more generalized models capable of adapting to different development environments and standards for requirements, pushing beyond single-project or mixed-projects instance sets. Alongside this, diversifying preprocessing choices can improve the quality of feature representation. Exploring other vectorization methods or preprocessing steps like advanced text normalization, domain-specific term extraction, or data augmentation could improve classification performance. One could also explore different standards for requirements definitions, expanding beyond the current scope to incorporate more complex requirement taxonomies or standards, such as User Stories.

In future work, a wider variety of parsers and encoders could be explored. This might include leveraging advanced readers such as the Stairs Reader, Spiders Reader, or Tree Reader, and even custom parsers for requirement extraction and encoding. Such advancements could improve the accuracy of initial representations, a crucial factor when inputting

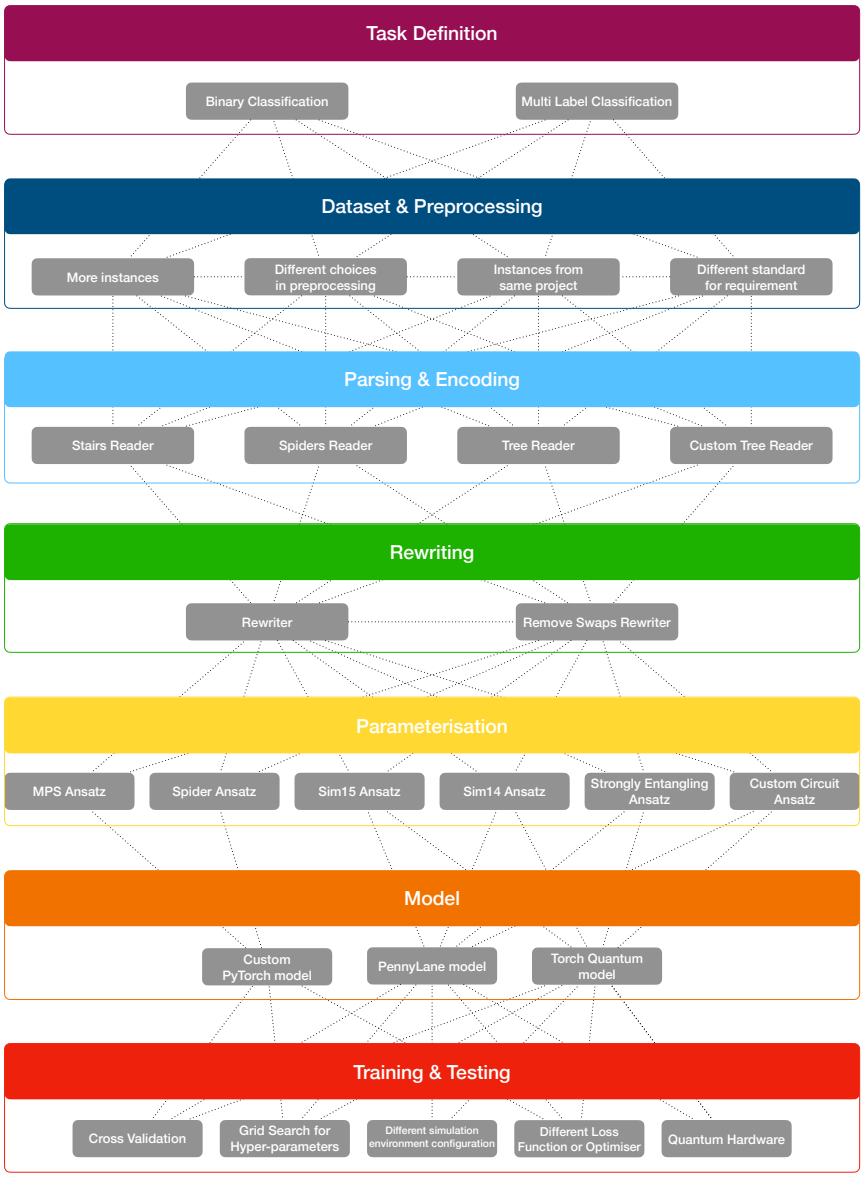


Fig. 14. Roadmap for unexplored application of Quantum NLP for the NFRs classification task.

data into quantum circuits or when parameterising such tensor networks. Exploring new quantum or classical encoding strategies may help align the peculiarities of requirements data with quantum computing's strengths.

In the rewriting phase, new strategies could be explored to optimize the string diagrams. While some Rewriters helped us to simplify string diagrams, future work could investigate other rewriting techniques or even custom rewriters that better optimize them to obtain simplified quantum circuits for specific backends. This would minimize noise and

errors during quantum operations, leading to more accurate results. More sophisticated quantum rewriting algorithms can also help circuits to execute more efficiently on simulators or actual quantum hardware.

One area of research that offers significant potential for improvement is circuit parameterization. Beyond the other available ansätze such as MPS and Spider ansatz, and Sim14, Sim15 and Strongly Entangled ansatz for quantum circuits, future work could experiment with custom circuit ansätze tailored for specific classes of NFRs or computational goals. For example, designing ansätze that encode NFR properties more efficiently could unlock new ways of achieving state-of-the-art performance in quantum simulations.

Expanding the choice of models is a clear future direction. In addition to the PyTorch models explored in this work, other frameworks such as PennyLane or Torch Quantum offer promising avenues for hybrid quantum-classical architectures. Future models might also incorporate fully custom models, which are fine-tuned to the particularities of requirement classification tasks. The development of more quantum-native models could push the boundaries of what quantum-enhanced learning can achieve.

Lastly, in the domain of training and testing, future studies should explore cross-validation techniques, particularly on larger datasets, to ensure that models generalize well to unseen data. Implementing grid searches for hyperparameter tuning can also lead to better-performing models. Additionally, configuring the quantum environment more thoroughly, including different compilation passes and shot configurations, could result in a better understanding of the effects that quantum probabilism has on performance. Moving beyond simulation, the ultimate goal would be running these models on actual quantum hardware, which could bring entirely new insights into the viability of quantum approaches for NFR classification. Furthermore, future experiments might examine alternative loss functions and optimizers, as different configurations could yield performance improvements.

These directions highlight the numerous potential for evolving the work presented, with a focus on expanding the dataset, exploring new models, parameterization techniques, and leveraging custom quantum architectures. By integrating these future advancements, the field can continue to push toward efficient, scalable, and accurate quantum-based solutions for NFRs classification. The continual refinement of task definition, model choices, and training strategies will shape the future of this intersection between quantum computing and software engineering.

6.3 Implications for RE Tasks Beyond Classification

While our empirical study targets the classification of NFRs, we believe the adoption of QNLP in RE holds broader potential, with implications that extend well beyond this initial use case. In particular, the decision to start with NFR classification was both strategic and pragmatic: it is a well-established, practically relevant, and well-bounded task that provides a suitable testing ground for evaluating the behavior of quantum-inspired models. Importantly, the structural and semantic foundations of QNLP, particularly its reliance on compositional semantics, entanglement, and superposition, indicate strong potential for tackling more complex RE challenges in the future.

Many RE activities require analyzing subtle semantic relationships between requirements, including synonymy, ambiguity, and contradiction. Quantum representations inherently capture these phenomena through entangled states, which can encode and manipulate multiple, overlapping meanings simultaneously. This capability is particularly relevant for conflict and inconsistency detection, a critical RE task where contradictions must be identified across different requirement sources. For example, a pair of requirements stating *"The system shall always encrypt user data"* and *"User data shall be accessible without delay"* introduces a potential trade-off between security and performance. Modeling such interactions may benefit not only from entanglement, which allows quantum circuits to represent dependencies between requirement fragments, but also from interference, a uniquely quantum phenomenon where

probability amplitudes reinforce or cancel each other out depending on their phase. Indeed, in natural language contexts, interference can be interpreted as a computational mechanism to amplify consistent or reinforcing semantic features and attenuate contradictory or conflicting ones. For instance, when two requirements encode conflicting goals (e.g., data encryption versus minimal latency), the corresponding quantum states could interfere destructively, reducing the likelihood of them being classified as jointly satisfiable. Conversely, requirements that are semantically aligned could exhibit constructive interference, reinforcing their compatibility. This interference-based signal modulation offers a way to model the degree of alignment or contradiction between requirements in a non-binary, probabilistic fashion, potentially capturing subtle nuances that classical vector-based similarity metrics often miss. As a consequence, interference allows quantum circuit-based models to natively encode and reason about semantic opposition, making them well-suited for detecting soft contradictions, unintended overlaps, or conceptual tension.

Another research opportunity is requirements traceability, where compositional models such as those supported by QNLP can align semantically related fragments across heterogeneous artifacts (e.g., linking a high-level goal to low-level specifications). The string diagram-based representations used in QNLP maintain syntactic and semantic structure, which could enable more principled and interpretable trace link generation and maintenance. Traditional NLP methods often rely on surface-level textual similarity or contextual embeddings to establish such links, which can miss deeper syntactic or semantic relationships, especially when the requirements are expressed in different levels of abstraction, granularity, or phrasing. In contrast, quantum natural language representations maintain explicit structural information via string diagrams, which encode not just word meanings but also their grammatical interactions using formal, rule-based constructions rooted in category theory. These diagrams can be systematically rewritten using operations such as cup removals, co-domain unification, or spider fusion, which simplify or normalize the structures without losing the underlying semantics. For instance, syntactically different requirements expressing the same intent (e.g., *"the system must ensure rapid access"* vs. *"quick response time is required"*) may be mapped to equivalent or transformable diagram structures through such rewrites, enabling a form of structural alignment that is more robust to linguistic variation than classical token-based methods. Furthermore, once transformed into quantum circuits, these diagrams can undergo parameterized transformations (e.g., tuning gate angles, modifying entanglement patterns) that preserve compositional meaning but adapt the representation to optimize downstream tasks like similarity scoring or alignment prediction. This pipeline thus supports a principled traceability mechanism, where semantically equivalent or related requirements can be identified through shared diagrammatic or circuit-based structure.

Our findings also suggest potential applications in requirement generation and refinement, where the objective is to derive, elaborate, or translate requirements from one form to another, e.g., transforming informal stakeholder statements into structured specifications. The compositional nature of QNLP models, rooted in category theory and tensor networks, makes them particularly well-suited for supporting such transformations. Because each component of a requirement is represented structurally and semantically within a string diagram, modifying or extending a requirement while preserving its core meaning can be systematically achieved through controlled diagram rewriting and circuit parameterization. This opens up possibilities for developing traceable refinement pipelines, where changes to requirements can be reasoned about at the level of their compositional semantics.

More in general, the representational strengths explored in this study, i.e., the integration of grammatical structure, the expressiveness of string diagrams, and the flexibility of quantum circuit parameterization, offer a foundation for a variety of RE tasks where semantic precision, structural alignment, and interpretability are essential. Tasks like requirements clustering, duplicate detection, or quality assessment (e.g., identifying vague or incomplete statements) could particularly benefit from these compositional modeling principles. This is especially relevant in low-resource or

domain-specific scenarios, where traditional NLP methods may struggle to generalize. Overall, our findings indicate that quantum-inspired representations have the potential to serve as a general-purpose, linguistically grounded framework for advancing a wide range of RE challenges. At the same time, our results reveal both opportunities and limitations that warrant further investigation. While we observed that quantum circuit-based models can offer competitive performance in some NFR classes and are particularly promising in capturing structural and semantic characteristics, they still lag behind traditional approaches based on pre-trained embeddings in terms of overall accuracy and stability. Moreover, the expressiveness of QNLP comes with increased complexity in model configuration and execution, especially under current hardware and simulation constraints. These findings suggest that, despite their theoretical appeal, the practical adoption of quantum-inspired representations in RE requires further empirical validation. As such, it is our hope that our findings may represent the foundation upon which further research may explore how these models behave in other RE tasks, under different data regimes, and with evolving quantum technologies. In doing so, the community can more clearly assess where and how QNLP provides an advantage over classical NLP techniques.

7 Conclusion

In this study, we explored the classification of NFRs using various computational approaches, ranging from word embedding solutions combined with traditional ML techniques to tensor network-based models and quantum circuit-based models. Our research aimed to assess how effectively these different approaches could handle the classification task and identify potential advantages offered by quantum computing in this domain.

We deliberately begin with a relatively simple and well-studied task, i.e.m requirements classification, to establish a baseline understanding of how QNLP models handle natural language in an RE context. Classification provides a structured and measurable evaluation, making it an ideal entry point before tackling more complex RE problems. In particular, our study serves as an initial step toward determining the potential impact of quantum computing in RE and identifying whether QNLP can offer a viable alternative or complementary approach to classical NLP techniques.

Through RQ1, we demonstrated that classical ML algorithms, when combined with word embeddings, provide high performance in classifying unseen NFRs. Techniques such as Word2Vec and GloVe achieved precision, recall, and F1-scores exceeding 0.80, indicating that these combinations remain highly effective for NFRs classification. However, techniques based on TF-IDF and BERT embeddings exhibited lower performance, revealing challenges in capturing meaningful representations of NFRs.

In RQ2, we took a first step towards using string diagrams to represent requirements, parameterizing them as tensor networks. Although the tensor network-based models did not outperform classical approaches in terms of overall metrics, the results demonstrated that this method could compete with traditional contextual embedding techniques. This approach opens up new avenues for exploring mathematical representations of language in software requirements.

Lastly, RQ3 examined the use of quantum circuit-based models for NFRs classification. Our results showed that while quantum simulations on classical hardware are still in the early stages, they offer some promise. In particular, the non-shot-based simulation produced a model capable of recognizing all types of NFRs, although its performance was lower than traditional tensor models. This indicates that quantum computing, despite its probabilistic nature and complex configuration, can be a valuable tool in NFR classification, albeit with further refinement and experimentation.

Note that the research undertaken here had a quantitative and exploratory focus, primarily aimed at assessing the overall performance of different computational approaches for classifying NFRs. This differs from a study with a more qualitative connotation, where the objective would be to assess the specific factors and configurations that influence performance. The goal in this phase was to explore the potential of different models, including shallow ML

algorithms, tensor network-based models, and quantum circuit-based models, to offer a comparative overview of their effectiveness. This is because our experiments were designed to primarily assess the impact of different requirement representations on classification performance. To ensure a fair comparison, we controlled several parameters while varying others. Our focus was to hold as an independent variable the requirement representation, i.e., the main variable under investigation, where we tested different embeddings (e.g., statistical embeddings, contextual embeddings, tensor-based quantum representations). As main controlled variables we have: (1) the dataset, as the same dataset was used across all experiments to eliminate dataset-induced variability; (2) preprocessing pipeline, indeed tokenization, normalization, and other preprocessing steps were kept consistent to prevent bias from text processing variations, (3) model hyperparameters, unless explicitly studied, key hyperparameters (e.g., learning rate, batch size) were kept constant to focus on the impact of the representation method; (4) evaluation metrics, as we used a consistent set of performance metrics (accuracy, precision, recall, F1-score) to enable meaningful comparisons.

While future studies, as outlined in our ongoing research agenda, aim to examine in more depth the pipeline configuration and its impact on NFR classification performance, we can already offer preliminary insights into the factors that may have influenced our results. For instance, the choice of embeddings in shallow ML approaches, the complexity of tensor contraction in string diagram representations, and the configuration of quantum circuit parameters (such as the number of shots, Hadamard layers, and rotations) are all likely contributors to the variability in performance observed. These factors, although not fully tested in isolation, suggest potential areas for optimization, and we anticipate that a more detailed investigation into these configurations will provide a clearer understanding of how to tailor each approach to this complex task. Future work will also include a systematic analysis of computation time and resource costs, both for quantum simulations and, when feasible, execution on real quantum hardware, to better assess the practical viability of quantum-inspired approaches. Another interesting research direction is to investigate performance variability across individual projects to evaluate the robustness and generalizability of quantum representations in cross-project scenarios.

References

[1] Mina Abbaszade, Mariam Zomorodi, Vahid Salari, and Philip Kurian. 2023. Toward Quantum Machine Translation of Syntactically Distinct Languages. arXiv:2307.16576

[2] David Ameller, Claudia P. Ayala, Jordi Cabot, and Xavier Franch. 2012. How do software architects consider non-functional requirements: An exploratory study. *Proceedings of 20th IEEE International Requirements Engineering Conference (RE)* (2012), 41–50.

[3] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. 1999. *Modern Information Retrieval.*

[4] Cody Baker, Lin Deng, Suranjan Chakraborty, and Josh Dehlinger. 2019. Automatic Multi-class Non-Functional Software Requirements Classification Using Neural Networks. *Proceedings of IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)* 2 (2019), 610–615.

[5] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. arXiv:1607.04606

[6] Andreas Borg, Angela Yong, Pär Carlshamre, and Kristian Sandahl. 2003. The Bad Conscience of Requirements Engineering : An Investigation in Real-World Treatment of Non-Functional Requirements. *Computer Science* (2003).

[7] Donald T Campbell and Thomas D Cook. 1979. Quasi-experimentation. *Chicago, IL: Rand Mc-Nally* (1979).

[8] Francesco Casillo, Vincenzo Deufemia, Fabio Palomba, and Carmine Gravino. 2024. A First Eye on Non-Functional Requirements Detection with Quantum NLP. https://drive.google.com/drive/folders/1tULvfFaa2tAJREf2HWpJzpNYvtWx2ntg?usp=sharing

[9] Stephen Clark. 2021. Something Old, Something New: Grammar-based CCG Parsing with Transformer Models. arXiv:2109.10044

[10] Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical Foundations for a Compositional Distributional Model of Meaning. arXiv:1003.4394

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805

[12] Carmine Ferrara, Francesco Casillo, Carmine Gravino, Andrea De Lucia, and Fabio Palomba. 2024. ReFAIR: Toward a Context-Aware Recommender for Fairness Requirements Engineering. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering (ICSE '24)*. ACM, Article 213. https://doi.org/10.1145/3597503.3639185

[13] Alessio Ferrari and Paola Spoletini. 2025. Formal requirements engineering and large language models: A two-way roadmap. *Information and Software Technology* 181 (2025), 107697.

[14] R. Guarasci, Giuseppe De Pietro, and Massimo Esposito. 2022. Quantum Natural Language Processing: Challenges and Opportunities. *Applied Sciences* (2022).

[15] Khan Mohammad Habibullah and Jennifer Horkoff. 2021. Non-functional Requirements for Machine Learning: Understanding Current Use and Challenges in Industry. *Proceedings of IEEE 29th International Requirements Engineering Conference (RE)* (2021), 13–23.

[16] Vojtěch Havlíček, Antonio D. Córcoles, Kristan Temme, Aram Wettroth Harrow, Abhinav Kandala, Jerry M. Chow, and Jay M. Gambetta. 2018. Supervised learning with quantum-enhanced feature spaces. *Nature* 567 (2018), 209 – 212.

[17] Valery Herrington and Oluwafemi Adedeji. 2023. At The Intersection of Medical Robotic Surgery and Drug Discovery with Quantum Computing. *Journal of Electrical Electronics Engineering* 2 (2023), 274–281. Issue 3.

[18] Neil Ireson, Fabio Ciravegna, Mary Elaine Califf, Dayne Freitag, Nicholas Kushmerick, and Alberto Lavelli. 2005. Evaluating Machine Learning for Information Extraction. In *Proceedings of the 22nd International Conference on Machine Learning*. ACM, New York, NY, USA, 345–352.

[19] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. FastText.zip: Compressing text classification models. arXiv:1612.03651

[20] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of Tricks for Efficient Text Classification. arXiv:1607.01759

[21] Dimitri Kartsaklis. 2016. Coordination in Categorical Compositional Distributional Semantics. In *Proceedings of the 2016 Workshop on Semantic Spaces at the Intersection of NLP, Physics and Cognitive Science*. 29–38.

[22] Dimitri Kartsaklis, Ian Fan, Richie Yeung, Anna Pearson, Robin Lorenz, Alexis Toumi, Giovanni de Felice, Konstantinos Meichanetzidis, Stephen Clark, and Bob Coecke. 2021. lambeq: An Efficient High-Level Python Library for Quantum NLP. arXiv:2110.04236

[23] Kamaljit Kaur and Parminder Kaur. 2023. BERT-CNN: Improving BERT for Requirements Classification using CNN. *Procedia Computer Science* 218 (2023), 2604–2611.

[24] Khaleduzzaman, Zarina Che Embi, and Ng Kok Why. 2023. A Systematic Review on Natural Language Processing and Machine Learning Approaches to Improve Requirements Specification in Software Requirements Engineering. *International Journal of Membrane Science and Technology* (2023).

[25] Fatemeh Khayashi, Behnaz Jamasb, Reza Akbari, and Pirooz Shamsinejadbabaki. 2022. Deep Learning Methods for Software Requirement Classification: A Performance Study on the PURE dataset. arXiv:2211.05286

[26] Eric Knauss, Siv Houmb, Kurt Schneider, Shareeful Islam, and Jan Jürjens. 2011. Supporting Requirements Engineers in Recognising Security Issues. In *Requirements Engineering: Foundation for Software Quality*, Daniel Berry and Xavier Franch (Eds.). Springer Berlin Heidelberg, 4–18.

[27] Debarshi Kundu, Archisman Ghosh, Srinivasan Ekambaram, Jian Wang, Nikolay V. Dokholyan, and Swaroop Ghosh. 2024. Application of Quantum Tensor Networks for Protein Classification. *bioRxiv* (2024).

[28] J. Lambek. 1999. Type Grammar Revisited. In *Logical Aspects of Computational Linguistics*, Alain Lecomte, François Lamarche, and Guy Perrier (Eds.). Springer Berlin Heidelberg, 1–27.

[29] Bing Li and Xiuwen Nong. 2022. Automatically classifying non-functional requirements using deep neural network. *Pattern Recognition* 132 (2022), 108948.

[30] Márcia Lima, Victor Valle, Estevão Costa, Fylype Lira, and Bruno F. Gadelha. 2019. Software Engineering Repositories: Expanding the PROMISE Database. *Proceedings of the XXXIII Brazilian Symposium on Software Engineering* (2019).

[31] Milda Maciejauskaitė and Jolanta Miliauskaitė. 2024. The Efficiency of Machine Learning Algorithms in Classifying Non-Functional Requirements. *New Trends in Computer Sciences* (2024).

[32] Vincenzo De Martino and Fabio Palomba. 2023. Classification, Challenges, and Automated Approaches to Handle Non-Functional Requirements in ML-Enabled Systems: A Systematic Literature Review. arXiv:2311.17483

[33] Maha A. Metawei, Mohamed Taher, Hesham Eldeeb, and Salwa M. Nassar. 2023. A topic-aware classifier based on a hybrid quantum-classical model. *Neural Computing and Applications* 35 (2023), 18803–18812.

[34] Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of 1st International Conference on Learning Representations*.

[35] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. arXiv:1310.4546

[36] Anderson Oliveira, João Lucas Correia, Wesley K. G. Assunção, Juliana Alves Pereira, Rafael Maiani de Mello, Daniel Coutinho, Caio Barbosa, Paulo Libório, and Alessandro Garcia. 2024. Understanding Developers' Discussions and Perceptions on Non-functional Requirements: The Case of the Spring Ecosystem. *Proc. ACM Softw. Eng.* 1 (2024), 517–538.

[37] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543. http://www.aclweb.org/anthology/D14-1162

[38] David M. W. Powers. 2011. Evaluation: from Precision, Recall and F-measure to ROC, Informedness, Markedness and Correlation. *Journal of Machine Learning Technologies* 2 (2011), 37–63.

[39] Qiskit Development Team. 2019. Qiskit: An Open-source Framework for Quantum Computing. *Zenodo* 10.5281/zenodo.2562111 (2019). https://doi.org/10.5281/zenodo.2562111

[40] Vinay R and Badari Nath K. 2023. Quantum Video Classification Leveraging Textual Video Representations. In *Proceedings of 4th International Conference on Communication, Computing and Industry 6.0*. 1–6.

[41] Abdur Rahman, Abu Bakar Siddik Nayem, and Saeed Siddik. 2023. Non-Functional Requirements Classification Using Machine Learning Algorithms. *International Journal of Intelligent Systems and Applications* (2023).

[42] K.M. Ashikur Rahman, Anwar Ghani, Sanjay Misra, and Arif Ur Rahman. 2024. A deep learning framework for non-functional requirement classification. *Scientific Reports* 14 (2024).

[43] Juan Enrique Ramos. 2003. Using TF-IDF to Determine Word Relevance in Document Queries. *Computer Science* (2003).

[44] M.A.F. Saroth, P.M.A.K. Wijerathne, and B.T.G.S. Kumara. 2024. Automatic Multi-Class Non-Functional Software Requirements Classification Using Machine Learning Algorithms. *2024 International Research Conference on Smart Computing and Systems Engineering (SCSE)* 7 (2024), 1–6.

[45] Sukin Sim, Peter D. Johnson, and Alán Aspuru-Guzik. 2019. Expressibility and Entangling Capability of Parameterized Quantum Circuits for Hybrid Quantum-Classical Algorithms. *Advanced Quantum Technologies* 2 (2019).

[46] James C. Spall. 1998. Implementation of the simultaneous perturbation algorithm for stochastic optimization. *IEEE Trans. Aerospace Electron. Systems* 34 (1998), 817–823.

[47] Jonas Stein, Ivo Christ, Nico Kraus, Maximilian Balthasar Mansky, Robert Müller, and Claudia Linnhoff-Popien. 2023. Applying QNLP to Sentiment Analysis in Finance. *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)* 02 (2023), 20–25.

[48] Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Neural Information Processing Systems*.

[49] Colin M. Werner. 2022. Towards A Theory of Shared Understanding of Non-Functional Requirements in Continuous Software Engineering. *2022 IEEE/ACM 44th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)* (2022), 300–304.

[50] Dominic Widdows. 2004. Geometry and Meaning. *Computational Linguistics* 32 (2004), 155–158.