UNIVERSITY OF SALERNO

Computer Science Department

Master's Degree in Computer Science

GRADUATION THESIS

# Exploring the Potential of Quantum NLP for Non-Functional Requirements Classification

SUPERVISOR
Prof. Fabio Palomba
CO-SUPERVISOR
Dr. Francesco Casillo
University of Salerno

CANDIDATE
**Marco Calenda**
0522501165

Academic Year 2022-2023

*This thesis was carried out at the*

sesa lab
SOFTWARE ENGINEERING
SALERNO

*To play a wrong note is insignificant; to play without passion is inexcusable.*

*- Ludwig van Beethoven*

## Abstract

*Requirements engineering* (RE) involves defining the services a system must provide and the constraints it must adhere to during development and throughout its lifecycle. This phase, often challenging due to evolving factors – such as changing customer needs or switching technologies – places a significant emphasis on *non-functional requirements* (NFRs), i.e., quality attributes across dimensions like ethics, infrastructure, and security, crucial for user satisfaction and sustaining market viability. Automating the identification and classification of NFRs from requirements documents, leveraging *machine learning* (ML) techniques, is essential for accelerating and enhancing the efficiency and quality of the entire RE process.

This thesis tackles the intricate task of classifying NFRs by exploring the possibilities offered by *quantum natural language processing* (QNLP), an emerging field that utilizes quantum computing principles for processing and analyzing natural language text. Toward a grammar-informed NLP, we leverage the DisCoCat framework to capture both distributional and compositional aspects of meaning. This study aims to assess the classification capabilities of the quantum approach compared to shallow ML models that utilize classical text vectorization techniques – such as BoW, TF-IDF and Word2Vec. Furthermore, the study aims to explore other quantum compositional models that are not grammar-based in order to assess the relevance of the grammatical structure in the context of NFRs specifications.

The empirical study conducted demonstrates the effectiveness of the DisCoCat framework in accurately classifying NFRs, showing that the grammar was an added value in this particular scenario. The comparison with classical language models reveals that quantum approach outperforms traditional approaches based on word frequency, while exhibiting similarities to Word2Vec, albeit with much fewer parameters. Overall, this research contributes to the understanding of QNLP's applicability in real-world scenarios and lays the foundation for future advancements in the field.

# Contents

# List of Figures

# List of Tables

# CHAPTER 1

# Introduction

## 1.1 Application Context

*Requirements engineering* (RE) is a process dedicated to the collection, documentation, and validation of the requirements and services a system is expected to provide [1]. Recognizing and identifying requirements within a system is not always a straightforward procedure, as they may express hidden characteristics or undergo frequent changes over time due to various factors, making their management even more challenging.

Among the requirements, a special attention is given to *non-functional requirements* (NFRs), which represent constraints imposed on a system, defining its quality attributes. Typically related to security, performance, and scalability, NFRs are crucial because they contribute to ensuring that the system meets user needs and remains competitive in the market. NFRs are often identified and specified relatively late in the development process, and sometimes they are managed implicitly in an undocumented manner [2]. However, the absence of a well-structured set of NFRs can lead to inappropriate software design and project failure [3].

## 1.2 Motivations and Objectives

The identification and categorization of requirements are often carried out by experts, whose task involves reading and understanding requirement specification documents through a manual labor that is very time-consuming [4]. The most typical form of representation and description of a requirement is natural language, which brings along all the difficulties related to text comprehension and interpretation. Given this difficulty, methods have been sought for the automatic classification of requirements by adopting various techniques of *natural language processing* (NLP).

In recent years, transformer-based *large language models* (LLMs), incorporating self-attention mechanisms [5] – such as BERT [6] – have demonstrated superior performance in NLP tasks compared to simpler architectures – e.g. recurrent neural network (RNN) [7] or more elaborated variants such as long-short term memory (LSTM) [8]. Despite their popularity and effectiveness, current LLMs have reached huge number of parameters, and they require considerable amount of data to function properly, leading to high complexity and open research on the interpretability of the learned information and the explainability of the model responses [9]. To address these issues, a *still-in-its-infancy* alternative consists in the exploitation of quantum computing for NLP tasks (QNLP) – a cutting-edge area gaining popularity and achievability in the last few years. The main idea is to model natural language and its underlying structure combining both (i) the *distributional* semantic [10], relying on statistics about the contexts in which words occur, and (ii) the *compositional* semantic, arguing that the meaning of a sentence is the result of the meanings of the words composing it and the grammatical rules used to combine them – a closer view to real human cognitive processes. In the field of QNLP, the *de facto* framework used to model natural language is the distributional compositional model of Coecke et al. [11] – often dubbed as DisCoCat – which allows encoding sentences as string diagrams, and takes steps towards a revolutionary *grammar-informed* NLP.

In this thesis, we explore the QNLP potential applied into a quasi-real scenario of RE, and in particular, the main objective of this research study is to comprehensively assess the effectiveness of the quantum approach in classifying NFRs compared to classical NLP solution. To carry this out, we investigate on which is the best text vectorization technique among *bag-of-words* (BoW), *term frequency–inverse document frequency* (TF-IDF), and *Word2Vec*, considering the best shallow ML algorithms at doing the task. We then compare the results with the ones yielded by DisCoCat in both classical and quantum implementation using tensor networks and quantum circuits, respectively. Furthermore, we conduct an empirical analysis on whether and to what extent the additional factor of grammar, considered by the DisCoCat framework, is beneficial for the task of classification within this specific application context. Therefore, we compare the distributional compositional model with linear readers which does not take into account the grammatical structure of the sentence, yet they are still quantum approach relying on the vectorial representation of entire sentences.

Given the limited capabilities of modern quantum devices, it is still challenging to scale a quantum model as a current LLMs and prove a practical quantum-advantage. As shown in recent experiments [12, 13], there are still many constraints, imposed by the number of required qubits and their stability, in terms of vocabulary range and sentences length, helding QNLP far from being used for real use cases. Nonetheless, this initial study plays a significant role in enhancing our comprehension of the process, technical difficulties, and the distinctive characteristics of this emerging computational paradigm.

## 1.3 Results Obtained

The results of the experiments showcasing promising overall results, emphasizing the efficacy of the DisCoCat model in acquiring meaningful sentence representations and accurately classifying NFRs. Classical implementation relying on tensor network performed similar to shallow ML employing frequency-based vectorization techniques – specifically BoW and TF-IDF – and its quantum counterpart achieved comparable results to Word2Vec, all while significantly reducing the number of parameters. In general, within the PROMISE dataset, security and operational requirements are the easiest to classify, while the greatest challenges are encountered for usability and performance requirements.

Furthermore, the DisCoCat-based model exhibited slightly superior performance compared to other linear models that do not consider the grammatical composition of sentences, despite still computing a tensor of the sentences. This observation suggests that, even though sentence simplifications were made for computational reasons, grammar remains a valuable aspect in this context, contributing to enhanced classification accuracy.

## 1.4 Thesis outline

This thesis follows a structured organization, as outlined below:

- *Chapter 2*: Introduces fundamental concepts related to requirements engineering and the NFRs classification. Also, it provides essential knowledge to about quantum computing, category theory, and the DisCoCat framework. Finally, it explores the state of the art concerning requirements classification QNLP.
- *Chapter 3*: Delves into the research questions and outlines the design of the empirical study conducted to assess the quantum approach.
- *Chapter 4*: Presents quantitative results and engages in a comprehensive analysis of the primary findings.
- *Chapter 5*: Discusses potential threats to the robustness and validity of the research and outlines the measures implemented to mitigate these challenges.
- *Chapter 6*: Serves as the conclusion of this thesis, summarizing the key contributions, discussing the implications of the findings, and presenting potential avenues for future research endeavors.

CHAPTER 2

# Background and state of the art

## 2.1 Theoretical Background

The core intention of this section is to to familiarize the reader with the theoretical underpinnings that form the basis of this research study. We begin by providing an introduction to requirements engineering and non-functional requirements classification, subsequently we delve into an exploration of common natural language processing techniques which are also employed in our research. Lastly, we provide a comprehensive explanation of quantum natural language processing focusing on category theory and the DisCoCat framework.

### 2.1.1 Non-Functional Requirements Classification

*Requirements engineering* (RE) is a systematic process that revolves around gathering and defining the services the system should offer and the constraints under which it operates and is developed. This phase encompasses evaluating the system's utility to the business through a feasibility study, finding requirements through *elicitation* and *analysis*, transforming these requirements into a standardized format (*specification*), and ensuring that the specified requirements align with the system envisioned by the customer through a *validation* process [1]. In summary, it is a comprehensive methodology ensuring that the developed system meets the needs and expectations of the end-user while aligning with the business objectives.

Within the software development lifecycle, RE is often considered one of the most challenging phase to execute, primarily because requirements are prone to evolving over time due to various factors such as changing customer needs, evolving technology, and shifting business priorities [14]. As such, the dynamic nature of requirements necessitates careful management and continuous communication between stakeholders to ensure that the system aligns with the evolving expectations and constraints.

Despite their heterogeneous nature, software requirements can be classified into two main types:

- *Functional requirements* (FRs): These pertain to the specific functionalities and features that a software system must possess to meet user needs and business objectives. Most of the time they are typically described in an abstract manner from the user itself, while more elaborated functional system requirements outline more technical aspects such as system functions, inputs, anticipated reactions to particular inputs, and expected output.

- *Non-functional requirements* (NFRs): These are quality attributes that the system should exhibit for achieving user satisfaction and sustain its market viability, encompassing various aspects – such as quality, ethics, infrastructure, security – as well as any constraints imposed on the application's development or the software development process itself. While defining all the types of NFRs may not be straightforward, the widely recognized repository for RE tasks, PROMISE [15], encompasses 11 subcategories of NFRs – each subcategory's type and description are elucidated in Table 2.1.

NFRs often hold greater significance than individual functional requirements, as users may find workarounds for functions that don't fully meet their needs, while failing to meet a NFR might render the entire system unusable. In fact, one of the most time-consuming and experienced task of RE is the classification of NFRs [4]. Unlike the functional counterparts, centrally written and well-formed in the requirement

**Table 2.1:** Descriptions of each type of NFR from PROMISE repository.

| Type | Description |
|---|---|
| Availability (A) | Describes how likely the system is accessible for a user at a given point in time. |
| Fault Tolerance (FT) | Degree to which a system, product, or component operates as intended despite the presence of hardware or software faults. |
| Legal & Licensing (L) | Certificates or licenses that the system must have. |
| Look & Feel (LF) | Describes the style of the product's appearance. |
| Maintainability (MN) | Degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers. |
| Operability (O) | Degree to which a product or system has attributes that make it easy to operate and control. |
| Performance (PE) | Performance relative to the amount of resources used under stated conditions. |
| Portability (PO) | Degree of effectiveness and efficiency with which a system, product, or component can be transferred from one hardware, software, or other operational or usage environment to another. |
| Scalability (SC) | Degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software, or other operational or usage environments. |
| Security (SE) | Degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization. |
| Usability (US) | Degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use. |

documents, NFRs are not always explicit, and most of the times, they are scattered and concealed into natural languages sentences that are ambiguous and imprecise. Furthermore, it has been demonstrated that the lack of knowledge of NFRs in the early stages has a huge impact on the total cost and the failure rate of IT projects [16, 3]. This is even worse for AI-enabled systems where specific NFRs – such as *bias* and *fairness* – might potentially discriminate against moral principles rooted in tradition, group values, or individuals, thereby undermining the company's reputation [17].

Requirements are commonly expressed through their textual representation, originating from documents or user reviews. While manual detection and classification offer a finer level of granularity and meaning, the manual process is labor-intensive and often impractical especially on large projects with a huge number of requirements. Additionally, conflicts, interactions, and dependencies often arise among different NFRs, adding further complexity to the entire process. For these reasons, automating the identification and classification of NFRs from requirements documents, facilitated by software assistance often employing machine learning techniques, holds substantial importance in enhancing the efficiency and quality of the RE process.

### 2.1.2 NLP for Requirements Classification

*Machine learning* (ML) is a branch of *artificial intelligence* (AI) which explores the study and construction of algorithms that can learn from data and make predictions based on it. ML can undertake various tasks based on the range of values it aims to predict, including:

- *Classification*: The goal of the model is to categorize or classify data into predefined classes or labels, yielding discrete outputs (e.g., identifying whether an image contains a cat or a dog).
- *Regression*: The model predicts a continuous outcome (e.g., predict the price of a house based on features like square footage, number of bedrooms, etc.).
- *Clustering*: The model groups data points into clusters or segments based on inherent similarities among them, without predefined categories (e.g., identifying groups of customers with similar interests).

In general, in the field of ML, there are three types of learning:

- *Supervised learning*: Models are initially guided during the learning process by providing already labeled data with their respective correct output to predict – a type of learning especially employed in classification and regression tasks.
- *Unsupervised learning*: Models are trained with data whose labels are hidden throughout the learning process. This implies the model's ability to extract common features and patterns from input data and create clusters on which predictions can be made.
- *Reinforcement learning*: Models are guided by rules and objectives, along with corresponding "rewards" and "punishments," which are useful for the learning process over times

*Natural language processing* (NLP) involves constructing models capable of performing tasks related to human-machine interaction, specifically the processing and analysis of data in natural language – such as text or speech. This field encounters various challenges due to the inherent ambiguity of natural language, lacks standardization (variations in spelling, grammar, and syntax) and context sensitivity.

One of the most common objective in the field of NLP is *language modeling* (LM), which involves creating models capable of determining the probability of a sentence appearing in a text, and this can be applied to various tasks, including:

- Translation: Dealing with the multiple possible translations of a word becomes more manageable by considering the highest probability in a specific context, e.g., determining the most probable translation of a word based on the context of the sentence.
- Correction: Knowing the probability of having the word "minutes" after "fifteen" can suggest the presence of an error in sentences like "The office is about fifteen minuets from my house."
- Speech recognition: Phrases reconstructed from a recording can be ambiguous or distorted. As such, comparing probabilities between two phrases like "I saw a van" and "eyes awe of an" can determine which is the correct phrase.

- Generation: Models are designed to create coherent and contextually relevant pieces of text. This task involves training models to understand the patterns and structures present in a given dataset and then generating new text based on that learned knowledge.

Another recurring challenge in NLP, which is also the focal point of this work, is *text classification* – i.e., assigning a binary (e.g., spam/non-spam email) or multi-class (e.g., categorizing scientific articles) label to a given text. As previously mentioned, a straightforward solution involves predefined rules, but this approach is time-consuming, therefore, ML techniques are commonly employed. To facilitate this process, textual data must undergo a step of *feature extraction*, wherein raw text is converted into a vectorized format that ML models can handle as input. Several methods are employed for text vectorization:

- *Bag-of-words (BoW)*: It considers a dictionary composed of all unique terms found within the corpus, and it utilizes the *term frequency* (TF), which is essentially the count of how many times a specific term appears. It is called a "bag" of words, because any information about the order or structure of words in the sentences is discarded. In the context of requirements classification, BoW represents a requirement, denoted as $j$, as a vector

  $$X_j = (x_{1,j}, x_{2,j}, \dots, x_{i,j}, \dots, x_{n,j}) ,$$

  where $x_{i,j}$ represents the weight of feature $i$ calculated by the TF of $i$ in the requirement $j$, and $n$ represents the total number of unique terms in the dictionary. To decrease the vector dimensionality $n$ one can choose to consider only the $k$ most frequent terms.

- *Term frequency-inverse document frequency (TF-IDF)*: Another widely used vectorization technique which considers (i) the raw TF, as BoW, and (ii) the *inverse document frequency* (IDF) for each term. We compute the document frequency $df_i$, namely the number of documents containing the term $i$, then applying logarithmic scaling on the result, we consider the inverse with $N$ the number of documents:

  $$idf_i = log_{10} \frac{N}{df_i}$$

$idf_i$ serves as a weight to distinguish highly frequent words present in all documents (e.g., "the," "it," etc.), which provide minimal information gain and may introduce bias. The TF-IDF score for term $i$ in the requirement description $j$, calculated as the following:

$$TF\text{-}IDF_{i,j} = tf_{i,j} \times idf_i$$

One of the drawbacks with TF-IDF (and BoW) is that it produces long and sparse vectors, meaning they have many null values because not all words appear in every document – this makes them challenging to use effectively as features for ML algorithms.

- *Word2Vec*: In response to the limitations of TF-IDF and BoW, it involves the use of dense vectors with few or no zero values and lower dimensionality to enhance generalization. It doesn't focus on word counts, instead it employs a regressor to predict the probability that two words might appear closely together within documents. This process allows Word2Vec to capture semantic relationships and context between words. After the learning phase, the regressor itself is discarded, and the learned model's weights are retained as the so-called *word-embeddings*. A common practice is to pre-train Word2Vec models on large text corpora to capture extensive language knowledge, then use this for downstream tasks. Word2Vec comprises two primary methodologies: the *continuous bag-of-words* (CBoW) model and the *skip-gram* model, each serving distinct roles in word-embeddings. CBOW predicts the central word within a context by considering its surrounding words, aiding in the comprehension of a word's meaning based on its context. In contrast, the skip-gram model reverses the task by predicting surrounding context words from a central word, facilitating tasks involving word influence on neighbors or generating contextually relevant word-embeddings.

### 2.1.3 Quantum NLP

#### Quantum Computing

*Quantum computing* leverages the principles of quantum mechanics to solve complex problems previously considered insurmountable for classical computers. Its applications span across various fields, including chemistry, cryptography, and machine learning, with the potential to disrupt traditional approaches and revolutionize problem-solving methods. Unlike classical computers, quantum computers use quantum bits or *qubits* – physical entities present in nature at the atomic and subatomic levels that can represent any two-state quantum-mechanical system.

While a classical bit can only take on the values 0 or 1, a qubit has the remarkable property of being, at any given time, in a *superposition* of both states simultaneously. Due to the probabilistic nature of the qubit measurements, the state of a qubit can not be observed, collapsing upon measurement to one of the computational basis states $|0\rangle$ or $|1\rangle$. The *Bloch sphere*, depicted in Figure 2.1, serves as a widely-used visual representation for a single qubit's state, its surface corresponds to any qubit pure state, with the two computational basis forming the north and south poles on the sphere.

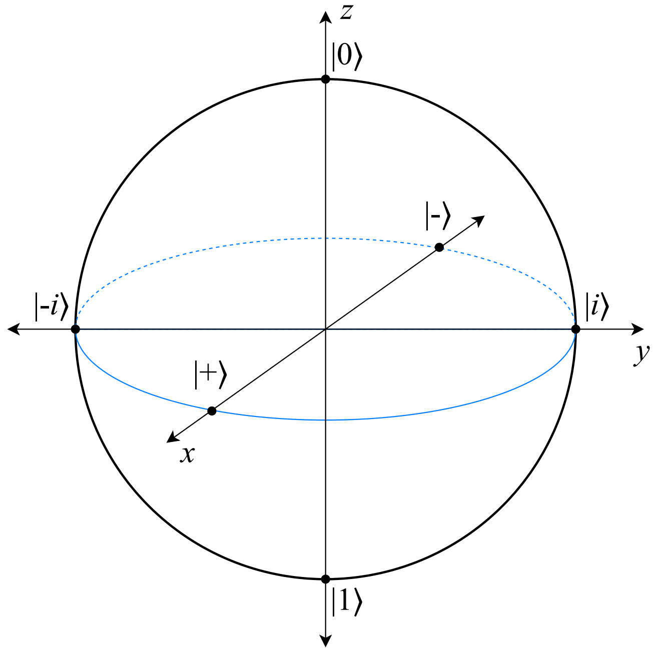


**Figure 2.1:** Bloch sphere of a qubit's state.

*Quantum circuits* act as physical implementations of quantum computing models and, at their core, there are *quantum gates* functioning analogously to logic gates in classical computers. In the context of this thesis, the most important quantum gates are represented by:

- *Pauli-X*: This gate performs a fundamental operation of interchanging the probabilities of collapsing into the computational basis – analogous to its classical counterpart, the *NOT* gate. On the Bloch sphere, the Pauli-X gate corresponds to a rotation about the X-axis. Similarly, the *Pauli-Y* and *Pauli-Z* gates represents respectively a rotation about the Y-axis and Z-axis.
- *Hadamard*: When a qubit undergoes a Hadamard gate it assumes an equal superposition state on the equator of the Bloch sphere, existing simultaneously as both $|0\rangle$ and $|1\rangle$ with equal probability.
- *Controlled-NOT*: A two-qubit gate that applies a Pauli-X operation to the target qubit only when the second control qubit is in state $|1\rangle$. By utilizing the Hadamard gate in conjunction with the CNOT gate, we are able to *entangle* qubits with unique and non-classical correlation, generating the so-called *Bell's states*.

Furthermore, it is important to note, as it becomes relevant for the upcoming chapters, that in a quantum circuit, the quantity of interest often depends only on the outcome distribution of a subset of qubits, while the remaining qubits have already yielded certain outcomes. This condition-driven selection process is referred to as *post-selection*. With this in mind, one needs to run the entire quantum circuit multiple times (*shots*), measuring all qubits; then, the data is restricted based on the condition being satisfied by the "post-selected" qubits.

### Quantum Machine Learning

As mentioned, the non-deterministic nature of quantum circuits implies that we need to run multiple shots to obtain a probability distribution encompassing all potential outcomes from which we can perform post-selection. In other terms, running a quantum circuit only once and using the output for some task would produce unreliable results. Despite the increasing promise shown by some near-term quantum devices, quantum processors contain a small number of qubits, and they still suffer from decoherence due to external environmental noise – as such, we talk about *noisy intermediate-scale quantum* (NISQ) devices. This types of hardware can only run low-depth gate sequences – this limitation arises because, without a fault-tolerance mechanism, every gate increases the error in the output.

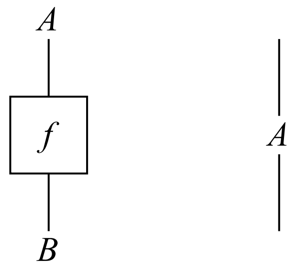
To address these limitations, *variational quantum circuits* (VQCs) have emerged as a dynamic and promising area of research, playing a key role in the field of *quantum machine learning* (QML). Denoted as $U(\theta)$, a VQC creates a quantum state, where $\theta$ represents adjustable parameters which are commonly input of rotations parameterized gates. Subsequently, measurements are taken on this quantum state, producing an expected value, and a loss function $L$ compares the QML model's predicted output with the expected one – this represents a degree of incorrectness.

Finally, a classical optimizer [18] iteratively executes this process with varying parameter assignments $\theta$, aiming to identify the parameters configuration that minimizes the loss function, thus, the quantum circuit yielding a value as close as possible to the target value. On real quantum hardware, the optimization proves to be computationally expensive, as such, it is used the simultaneous perturbation stochastic approximation (SPSA) [19] which relies on a numerical gradient estimate derived from a prior loss function measurement, avoiding the need for analytical gradient circuit evaluations.

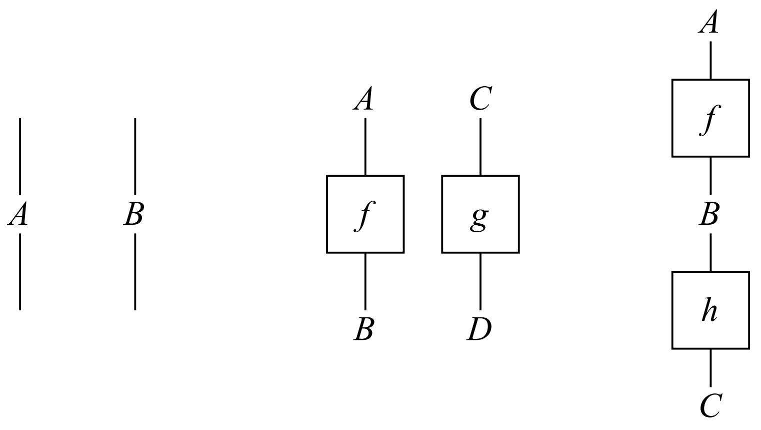
### Category Theory

The connection between quantum theory and NLP lays its foundations in *category theory*, a fundamental branch of mathematics that offers a versatile framework for exploring the connections and relationships between diverse mathematical structures. Furthermore, this framework comes with a diagrammatic calculus that simplify the categorical computations and also applies to computations in the upcoming vector space analog.

A *category* comprises a collection of *objects*, denoted as $A, B, C, \ldots$, and $f : A \to B, g : C \to D, h : B \to C, \ldots$ a collection of *morphisms* connecting these objects. Each object $A$ has the identity morphism $1_A : A \to A$. Morphisms are depicted by boxes and objects by lines. For instance, the morphism $f : A \to B$ and object $A$ are depicted as follows:



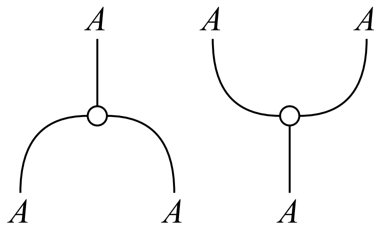
Morphisms with matching types can be composed, e.g, $f \circ h$ performs $h$ after $f$ since the resulting type $B$ of $f$ is also the initial type of $h$. We define *monoidal category* a category equipped with the *monoidal product* ($\otimes$) and the *monoidal unit* $I$ in order to compose morphisms "in parallel", and also to conceive two objects $A$ and $B$ as one whole. As an example, the combined object $A \otimes B$, and the morphisms $f \otimes g$ and $f \circ h$, considering morphisms from the previous example, are depicted as follows:



A monoidal category is *symmetric* if for any two objects $A$ and $B$, we have the natural isomorphism $A \otimes B \cong B \otimes A$. In the context of symmetric monoidal category, a *Frobenius algebra* provides for each object $A$ a commutative operation implementing the so-called *spider*, or formally, the morphisms $\Delta$ and $\mu$

$$\Delta : A \to A \otimes A \qquad \text{and} \qquad \mu : A \otimes A \to A \,,$$

which have a diagrammatic form as follows:



*Rigid categories* are monoidal categories where each object $A$ has both a left adjoint $A^l$ and a right adjoint $A^r$, along with the following morphisms:

$$A \otimes A^r \xrightarrow{\epsilon^r_A} I \xrightarrow{\eta^r_A} A^r \otimes A \,, \qquad A^l \otimes A \xrightarrow{\epsilon^l_A} I \xrightarrow{\eta^l_A} A \otimes A^l \,.$$

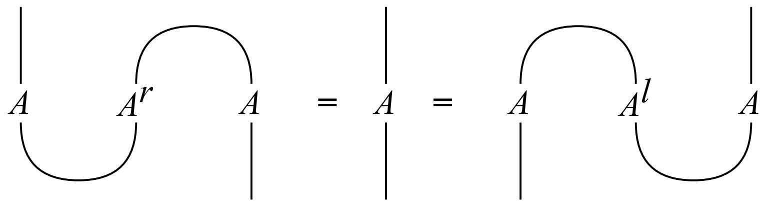
These structural morphisms between an object and its adjoints are called *cups* ($\epsilon$-map) and *caps* ($\eta$-map) because they are depicted as bent wires in the diagrammatic notation. They are also subject to the so-called *"snake equations,"* namely the following equalities:

$$(1_A \otimes \epsilon^l_A) \circ (\eta^l_A \otimes 1_A) = 1_A \qquad (\epsilon^r_A \otimes 1_A) \circ (1_A \otimes \eta^r_A) = 1_A$$
$$(\epsilon^l_A \otimes 1_A) \circ (1_{A^l} \otimes \eta^l_A) = 1_{A^l} \qquad (1_{A^r} \otimes \epsilon^r_A) \circ (\eta^r_A \otimes 1_{A^r}) = 1_{A^r}$$

In order to satisfy the aforementioned snake equations, wires are visually *"yanked"* resolving into an identity wire. To showcase the clarity enhancement and flexibility of this notation, the diagram for

$$(\epsilon^r_A \otimes 1_A) \circ (1_A \otimes \eta^r_A) = 1_A = (1_A \otimes \epsilon^l_A) \circ (\eta^l_A \otimes 1_A)$$

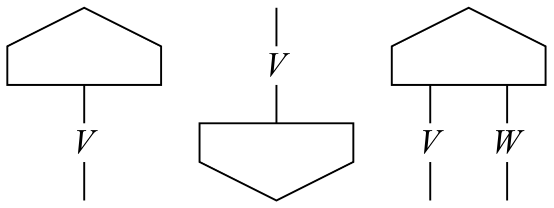
is the following:



A symmetric rigid category is defined *compact closed category*, and as a result of the symmetry we have $A^l = A^r = A^*$. The category of vector spaces, *FVect*, also constitute a compact closed category considering finite dimensional vector spaces $V, W, \ldots$ as objects, linear maps $f : V \to W, \ldots$ as morphisms and tensor product as monoidal product. Elements of a vector space $V$, i.e., vector $\vec{v} \in V$, are morphisms defined as $\vec{v} : \mathbb{R} \to V$. Similarly, *FHilb* is a compact closed category with Hilbert spaces as objects and unitary maps between Hilbert spaces as morphisms. For instance, the vectors

$$\vec{v} : I \to V \qquad \vec{w} : V \to I \qquad \vec{z} : I \to V \otimes W \,,$$

referred to effects ($\vec{w}$) and states ($\vec{v}, \vec{z}$) in quantum theory, are depicted as follows:



The extensive analogies and compatibilities observed between quantum theory and category theory have led to the emergence of a field known as *categorical quantum mechanics* (CQM). In this study, we leverage CQM for NLP, and to achieve this, we need a parallelism between sentences and categories. This connection is found in *pregroup grammars*, introduced by Lambek [20]. In this framework, cups $\epsilon_p^r$ and $\epsilon_p^l$ serve as grammar reductions to perform the sentence *parsing*, and are represented as

$$p \cdot p^r \to 1 \qquad \text{and} \qquad p^l \cdot p \to 1 \,.$$

For instance, given types $n$ for nouns and $s$ for sentences, a transitive verb would have a type of $n^r \cdot s \cdot n^l$, indicating that it expects a noun on the left and another noun on the right to form a sentence. Therefore, the derivation for the sentence *"Developers create software"* is

$$\begin{aligned} n \cdot (n^r \cdot s \cdot n^l) \cdot n &\to (n \cdot n^r) \cdot s \cdot (n^l \cdot n) \\ &\to 1 \cdot s \cdot 1 \\ &\to s \end{aligned}$$

The single type "$s$" at the end shows that the sentence is grammatically correct according to the pregroup grammar rules.

## DisCoCat: a Quantum-inspired NLP Framework

The ***Dis****tributional,* ***Co****mpositional,* ***Cat****egorical* (DisCoCat) framework, introduced by Coecke et al. [11], allows a transition from string diagrams in the category of pregroups derivation to tensor networks or quantum circuits in the category FVect or FHilb, respectively. This is achieved by a mapping that sends pregroup types (and relative adjoints) to vector space ($n \to N$ and $s \to S$), composite types to tensor product spaces (e.g. $n^r \cdot s \cdot n^l \to N \otimes S \otimes N$), and convert pregroup reduction to tensor contraction. This approach allows us to determine the meaning of a sentence, particularly, the compositional meaning is derived by interpreting the pregroup derivation of the sentence as a map and then applying it to the tensor product of the distributional meanings of the words within the sentence.

For instance, focusing on the classical case of FVect, the meaning of *"Developers create software."* is

$$(\epsilon_N \otimes 1_S \otimes \epsilon_N) \circ (\overrightarrow{Developers} \otimes \overrightarrow{create} \otimes \overrightarrow{software}) \ ,$$

and it is depictable via the string diagram in Figure 2.2 where each box corresponds to a tensor, with its order determined by the number of wires (based on the space of the pregroup type), while the cups indicate tensor contractions.

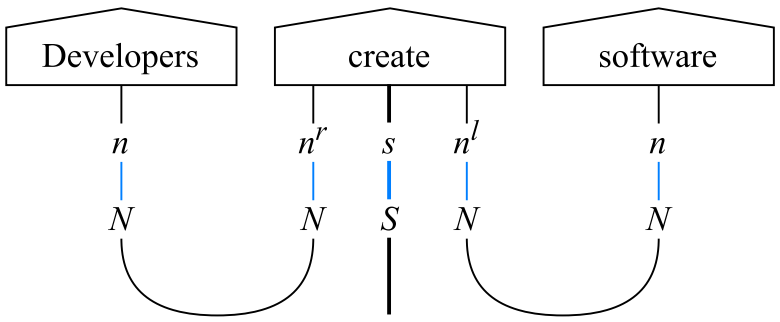


**Figure 2.2:** String diagram mapped to vector space.

It's important to emphasize that in this context, the framework still adhere to the widely accepted distributional hypothesis considering words as word-embeddings. However, they do exist within separate dimensional vector spaces rather than a singular unified space, and their interconnections are established through pregroup mapping instead of solely adhering to the conventional frequency-based approach or more intricate techniques like attention mechanisms.

In the context of quantum computing, we can readily adapt this approach to FHilb, by representing each individual word as a pure state $\psi_{word}$ in the finite-dimensional Hilbert space associated to its type. These quantum states are subsequently tensor-producted together and subjected to a map in which the pregroup grammatical structure is realised with entangling effects $\langle Bell|$. As a result, the previous example sentence becomes the following:

$$(\langle Bell| \otimes \mathbb{I} \otimes \langle Bell|) \circ (\psi_{Developers} \otimes \psi_{create} \otimes \psi_{software}) \ .$$

Qubits inherently represent tensor product states, in particular, $n$-dimensional classical vectors can be stored in $log_2 n$ qubits so they provide an efficient way to implement this algorithm with linear scalability concerning input size. In Table 2.2 is shown the practical advance in terms of storage of noun-verb-noun sentences, assuming a basis of size 2000 for both the vector space $N$ and $S$ [21].

**Table 2.2:** Storage comparison between quantum and classical computers.

| | **1 sentence** | **10k sentences** |
|---|---|---|
| Classical | $8x10^9$ bits | $8x10^{13}$ bits |
| Quantum | 33 qubits | 47 qubits |

## 2.2 Related Works

In this section we delve into an exploration of existing literature and the current landscape in the field of NFRs classification and QNLP synthesizing prior research and highlighting key advancements, methodologies and findings.

### 2.2.1 NFRs classification

Lu et al. [22] extracted the requirements from the users' opinions about some apps in Google Play and AppStore, and classified them into four types of NFRs, i.e., reliability, usability, portability, and performance. Their approach combines four vectorization techniques (BoW, TF-IDF, $\chi^2$, and the proposed AUR-BOW) with three shallow ML algorithms (NB, J48, and Bagging) to classify the reviews. The combination of AUR-BOW with Bagging achieved the best result (f1-score: 71%).

One of the challenges faced when conducting researches in this area is the limited availability of public databases containing software requirements. One of the most frequently utilized repository for software requirements is the PROMISE repository [15] – their NFR dataset comprises 11 classes with 625 labeled requirements from 15 projects. To address labeling issues, Dalpiaz et al. [23] manually annotated PROMISE dataset as well as requirements from 7 other projects. Nevertheless, there is still a significant limitation in training data, which often results in models overfitted to the dataset, thus, they lack the ability to generalize effectively to unseen projects.

In response to these challenges, Hey et al. [24] introduced a novel approach called NoRBERT, which leverages Transfer Learning (TL), i.e., an emerging strategy in which a system developed for one task is repurposed for a related but different task. By fine-tuning the BERT language model, they enhanced the performance of NFRs classification up to F1-score of 90% for FR/NFR classification. Casillo et al. [25] present an approach to automatically detect privacy requirements in user stories. Their methodology involves using NLP techniques to extract privacy-related information from user stories, and a pre-trained Convolutional Neural Network (CNN) to process this information. To improve privacy disclosure detection, they utilize a TL technique, showcasing its versatility in addressing various aspects of NFRs classification.

### 2.2.2 QNLP

As previously stated, in the present NISQ era, the endeavor to implement a quantum approach that can outperform state-of-the-art LLMs in real-world tasks presents a notably diverse set of challenges. To the best of our knowledge, known quantum algorithms with theoretically proven speedups rely on fault-tolerant quantum computers or QRAM – i.e., an efficient way to load classical data on a quantum computer – both of which are currently unavailable.

The pioneering theoretical contributions made by Zeng and Coecke [21] utilized the DisCoCat model to achieve a quadratic speedup in sentence classification tasks, while subsequent theoretical work by Coecke et al. [26] laid the foundation for implementations specifically on existing NISQ devices.

Concerning practical experiments, Meichanetzidis et al. [12] provided a groundbreaking proof of concept that concrete QNLP is feasible within the NISQ era. They achieved this by successfully running and optimizing a classifier on IBM quantum computers using a dataset consisting of 16 artificial short sentences. Building upon this foundation, Lorenz et al. [13] took the natural next step, presenting two NLP experiments involving medium-scale binary classification with 130 artificial sentences generated by a simple context-free grammar. Their results demonstrated a smooth convergence of models in both simulated and quantum hardware runs. Martinez and Leroy-Meline [27] expanded on this work by conducting a multi-label sentiment analysis experiment on a much larger dataset comprising over 800 artificial sentences – their research showcased the scalability of such a quantum framework.

Taking into account the challenges that have arisen and the fact that research advancements have only recently emerged, there are no QNLP application concerning NFRs classification. However, in the realm of classical ML solutions, this task has been a focal point for researchers for over a decade, leading to the development of numerous approaches. Employing lexical, syntactical and meta-data features, Kurtanović and Maalej [28] developed and evaluated a binary and multi-class classifier based on Support Vector Machine (SVM) algorithm for identifying usability, security, operational, and performance NFRs. Their binary classifiers achieved f1-score ranging between 73% and 90%, while the multi-class classifier between 70% and 82%.

CHAPTER 3

# Research Methodology

In this chapter, we outline the design of the empirical study we conducted. First, we introduce the research questions and the rationale behind their formulation. Subsequently, we delve into the description of the data used for our analysis, the QNLP pipeline involved, and the validation methods employed. Lastly, we elucidate the criteria we adopted for evaluating our ML model, along with a discussion on potential threats to validity. All the generated files can be found within the GitHub repository [1].

## 3.1 Research Questions

In the course of this research, we have adopted the Goal-Question-Metric (GQM) approach [29], a well-established and rigorous methodology that provides us with a systematic framework for conducting our study. GQM assists in formulating clear goals, refining specific research questions (**RQs**), and establishing relevant metrics to evaluate our progress and outcomes. This methodology not only enhances the rigor of our study but also contributes to maintain a focused and goal-oriented research trajectory, ultimately leading to more meaningful and insightful findings.

[1]https://github.com/MCalenda/Master-Thesis

We first introduce our main goal as the following:

◎ **Our Goal.** The main goal of this study is to provide a practical view of how a quantum model relying on string diagrams can be used to model domain-specific sentences and train a classifier applied for a real scenario task of NFRs classification.

To assess the effectiveness of the quantum model in accurately classifying NFRs, we first investigate on which is the best shallow ML algorithm at doing the task using the well established text vectorization techniques introduced in Chapter 2. Taking into account the motivation introduced in the previous chapters, we are not considering LLMs – such as BERT – as they are not in the scope of this comparison considering their huge difference in number of parameters. We assess both the binary and the multi-class capabilities of the NFRs classifier.

Then, we aim at compare the best shallow ML algorithms with the DisCoCat framework, and in particular its classical implementation in which the abstract representation given by a string diagram is directly translated into a tensor network and executed on classical hardware. As such, this led to the formulation of our first research questions (**RQ**$_1$ and **RQ**$_2$):

🔍 **RQ**$_1$**.** *To what extent can shallow ML algorithms classify the four most frequent NFR subcategories in the PROMISE dataset: usability, security, operational, and performance?*

🔍 **RQ**$_2$**.** *Is the classical implementation of DisCoCat accurate at least as the best shallow ML algorithms in classifying the four most frequent NFR subcategories in the PROMISE dataset: usability, security, operational, and performance?*

Subsequently, as first proof-of-concept of the quantum potential, we aim at translate string diagrams into quantum circuits and execute exact (non shot-based) simulations on classical hardware. Therefore, we define our third research question (**RQ**$_3$) as the following:

🔍 **RQ**$_3$**.** *Is the quantum implementation of DisCoCat accurate at least as the best shallow ML algorithms in classifying the four most frequent NFR subcategories in the PROMISE dataset: usability, security, operational, and performance?*

Lastly, we asked as fourth research question (**RQ**$_4$):

🔍 **RQ**$_4$**.** *To what extent does the grammatical structure of sentences impact NFRs classification when following the quantum approach?*

Through **RQ**$_4$, we aim at provide an empirical analysis on the pros and cons of considering the compositional aspect of meaning – i.e., the pregroup derivation – despite a full-distributional approach. In other terms, we aim to assess how does the DisCoCat framework compared to other lightweight compositional model, widely used in QNLP, which do not take into account the grammar of the sentences.

## 3.2 Data Collection

To conduct the evaluation, we used the PROMISE NFR dataset [15], and in particular a relabeled and extended version provided by Dalpiaz et al. [23], a widely used dataset in the research area of software requirements classification. It consists of 971 requirement descriptions, collected from 48 different projects, among those 446 are FRs and 525 are NFRs. The full distribution of the NFR subcategories is shown in Table 3.1 – as shown, certain classes within the dataset are not adequately represented. Specifically, the NFR categories of portability (PO), fault tolerance (FT), and legal (L) are quite rare in the dataset. Consequently, this study concentrates on the four most well-represented NFR categories: usability (US), security (SE), operational (O), and performance (PE).

First, we converted the original NFRs dataset to a CSV file, and we filtered out from the dataset the FRs and the NFRs not in the scope of the research questions. In Figure 3.1 is visually showed the distribution of the NFR subcategories considered in this work. The dataset exhibits a slight imbalance among security requirement categories, and the idea of applying undersampling to the majority class has been dismissed to prevent excessive information loss. Therefore, precautions need to be taken during the training phase.

Despite its limited size, which may be considered trivial for a robust LLM, the dataset used in this study presents a challenging task for a quantum approach – especially the DisCoCat framework. In order to reduce the vocabulary size and

**Table 3.1:** Full distribution of NFR subcategories in PROMISE NFR dataset.

| Type | # Requirements | Percent |
|---|---|---|
| Functional | 446 | 45.9% |
| Availability (A) | 31 | 3.1% |
| Fault Tolerance (FT) | 18 | 1.8% |
| Legal (L) | 15 | 1.5% |
| Look & Feel (LF) | 49 | 5.0% |
| Maintainability (MN) | 24 | 2.4% |
| Operational (O) | 77 | 7.9% |
| Performance (PE) | 67 | 6.9% |
| Portability (PO) | 12 | 1.2% |
| Scalability (SC) | 22 | 2.2% |
| Security (SE) | 125 | 12.8% |
| Usability (US) | 85 | 8.7% |
| **Total** | **971** | **100%** |

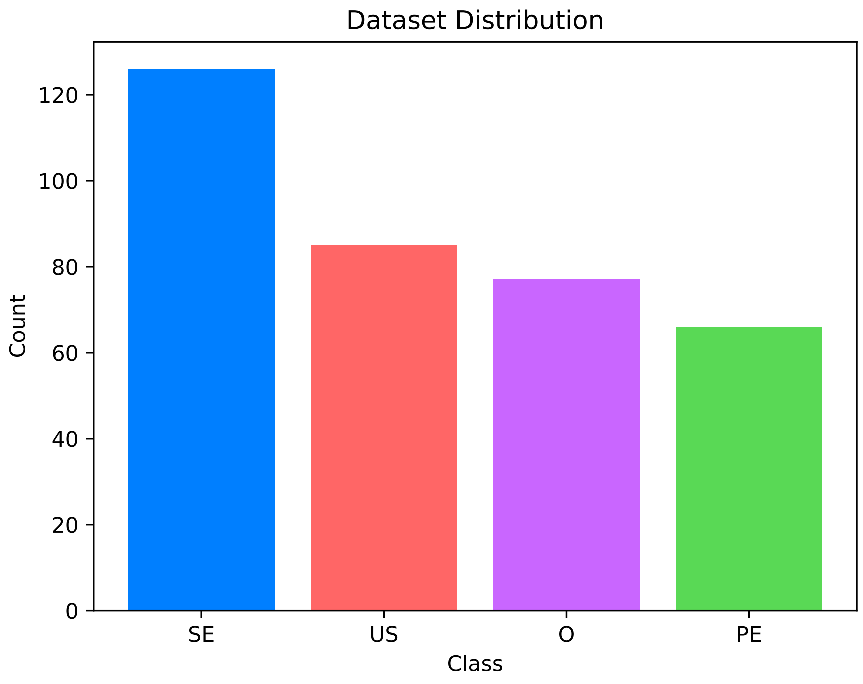


**Figure 3.1:** NFR subcategories considered in this study.

grammatical complexity, we performed a manual filtering, modifying sentences composed by (i) words overly specific or technical which appear infrequently in the dataset, (ii) large requirement descriptions composed by more than one sentence and (iii) sentences with a very complex or wrong grammatical structure which give error during parsing. The aim is to produce string diagrams with simple and accurate compositional semantics, for instance, the following PE requirement has been discarded:

*The product shall provide Dynamic Change Support and Transparent resource addition. The product shall support Transparent Resource addition and Dynamic Change support to provide scalability and avoid service interruptions of more than 1 day.*

After manually filtering the dataset, our dataset still presents unique challenges compared to datasets used in analogous experiments in the literature, as it comprises 322 words appearing twice or less (making it difficult to gain a deep semantic representation) and possessing an average sentence length of 8 words. To gain a deeper insight into the requirements within our final dataset, the following example sentences exemplify a SE and a O requirement, respectively:

*The system shall prevent malicious attacks including denial of service.*

*The product must be able to interface with any HTML browser.*

## 3.3 Data Preprocessing

In order to prepare the requirement text for analysis we use various preprocessing techniques developed with the aim of transforming raw text into a machine-readable format. This process is called *normalization*, and the main steps performed and their implications are outlined below:

- *Tokenization*: we break down the requirement description into individual components called *tokens*. In our case, we performed tokenization at word level throwing away certain characters – such as punctuation.
- *Stop-words removal*: some of the most common words in English are removed as they do not carry much meaning. Instances of stop-words encompass determiners (e.g., "a", "the"), auxiliary verbs (e.g., "am", "was"), prepositions (e.g., "in", "at"), conjunctions (e.g., "and", "but", "or") and digits.
- *Lemmatization*: we reduce words to their base form or lemma (dictionary form). This can result in more accurate analysis, as it takes into account the context in which the word is used. The aim is reduce the number of unique words in order to improve the efficiency of subsequent processing, in our case, it plays a key role since we are converting each unique word into a tensor.

It's crucial, since we are also employing the DisCoCat framework, to remove any word inflection but respect their part-of-speech in the sentence – a not so relevant issue in a classical NLP preprocessing pipeline. As a result, certain normalization steps are omitted when employing the quantum approach, as they may lead to errors or misleading parsing, and they are managed in the subsequent diagram rewriting phase.

## 3.4 QNLP pipeline

In this section, we provide a detailed walk-through of the sequential steps within our QNLP pipeline, which draws inspiration from the one proposed by Kartsaklis et al. [30] for the *Lambeq* library [2], a Python-based toolkit designed for QNLP. Throughout our research, we extensively utilize Lambeq's functionalities at various stages of the pipeline – for a holistic view of the entire process refer to Figure 3.2.

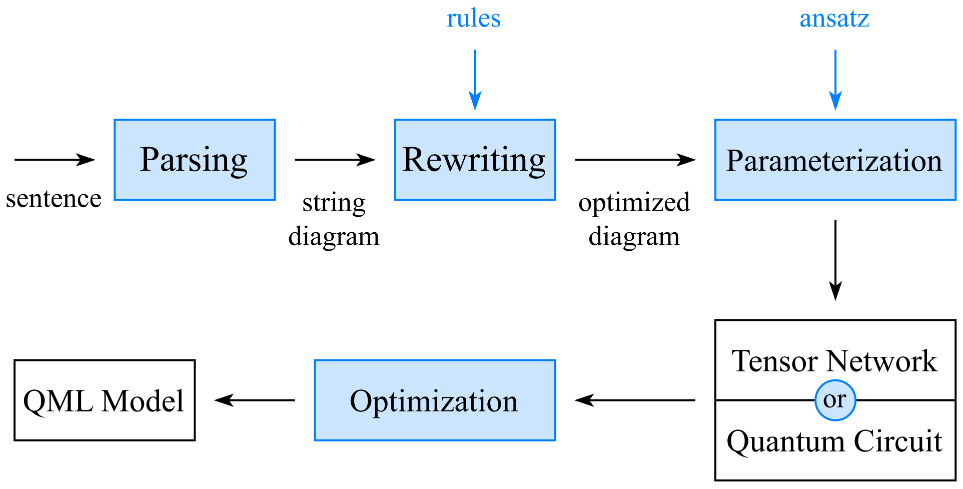


**Figure 3.2:** General QNLP pipeline.

### 3.4.1 Parsing

When involving DisCoCat as compositional model, a syntax tree needs to be provided by a statistical parser but, as of the current state of research, there is no pregroup parser that is capable of producing derivations for all types of sentences [30]. To address this limitation, Yeung and Kartsaklis [31] proposed an alternative version of DisCoCat that leverages a *Combinatory Categorical Grammar* (CCG) [32], which is positioned within the Chomsky hierarchy between context-free and context-sensitive grammars – a balanced compromise between expressive power and computational complexity. In our study, we adopt *Bobcat CCG* to parse the sentence and construct a syntax tree, which is then mapped into a string diagram that faithfully encodes the syntactic structure of the sentence.

[2]https://github.com/CQCL/lambeq

Concerning the targets, i.e. the label to predict, we utilize a *one-hot encoding* scheme as showed in the following snippet where is performed sentence parsing. This representation is particularly convenient for our models since the quantum measurement output corresponds to a vector of probabilities for the measured qubit.

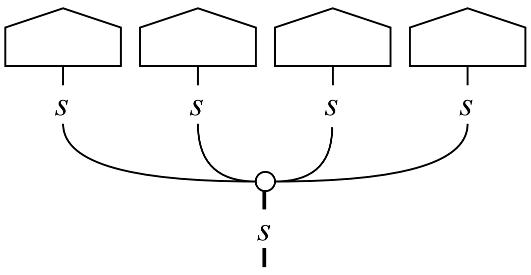
```python
def create_multi_diagrams(df, reader):
    diagrams, targets = [], []
    for _, row in enumerate(df.to_numpy()):
        sentence, target = row[0], row[1]
        diagrams.append(reader.sentence2diagram(sentence))
        if target == "US":
            targets.append([1.0, 0.0, 0.0, 0.0])
        elif target == "SE":
            targets.append([0.0, 1.0, 0.0, 0.0])
        elif target == "O":
            targets.append([0.0, 0.0, 1.0, 0.0])
        elif target == "PE":
            targets.append([0.0, 0.0, 0.0, 1.0])
    return diagrams, targets

# reader = SpidersReader
reader = BobcatParser(root_cats=['S'])
diagrams, targets = create_multi_diagrams(df, reader)
```

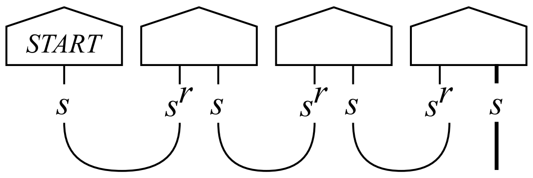
However, DisCoCat is not the sole option, as multiple approaches exist towards parsing and Lambeq offers an alternative solution by incorporating the concept of *linear reader* which focuses solely on the "*s*" pregroup type. By doing so, it can effectively handle informal and ungrammatical text without the burden of strict grammatical constraints giving more beneficial in scenarios where maintaining high parsing accuracy is crucial. We employ linear readers to address **RQ**$_4$ in order to assess the contribution of grammatical structure to the task of NFRs classification. The main linear readers are briefly explained below:

- *Spiders reader*: this approach employs a BoW strategy, representing a sentence as a multiset of words without taking into account the order of words or any other syntactic relationship between them (Figure 3.3a). To carry this out, exploiting the Frobenius algebra, the reader connects words using the spiders morphisms.

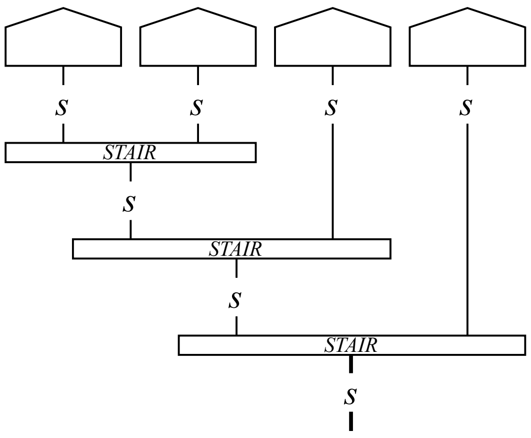
- *Cups reader*: a word-sequence approach consisting in a simple tensor network in which all tensors have the same shape and each is connected to the next one with a cup (Figure 3.3b). The string diagram incorporates a *"START"* symbol at the beginning, which is depicted as an order-1 tensor. This guarantees that the outcome of the computation, representing the sentence, will also be a order-1 tensor.
- *Stairs reader*: takes a recurrent approach to combine consecutive words using a box, which resembles the functioning of a RNN. This allows to capture sequential dependencies in the sentence structure effectively (Figure 3.3c).



**(a)** Spiders reader



**(b)** Cups reader



**(c)** Stairs reader

**Figure 3.3:** Linear readers included in lambeq.

### 3.4.2 Rewriting

As we proceed towards constructing quantum circuits, it becomes necessary to optimize our diagrams in order to make the best use of the limited quantum resources. Considering that syntactic derivations in pregroup form can become overly complex, especially in our study which deals with real-life examples of requirements descriptions that may include elaborate and detailed specifications, we employ the so-called *rewriting rules*. Rewriting is an optional but crucial step in the pipeline, enabling the simplification and optimization of the string diagram structure while preserving its expressiveness. This process is essentially a structure-preserving transformation, made possible by our utilization of the compact closed categories FVect and FHilb providing additional degrees of freedom.

Most of the rewriting rules pertain to the internal linking of words using caps to "bridge" disconnected wires, altering the regular flow of information and compositional connections, thereby simplifying the tensor dimensionality. After applying the rewriting rules, the diagram is typically further normalized by "yanking" the wires using the snake equations. The rewriting rules involved in this phase are:

- *Connector rule*: sentence connectors such as "that" can be eliminated using the connector rule and replaced with caps.
- *Determiner rule*: remove determiners, e.g. "the", by replacing them with caps.
- *Auxiliary rule*: this rule involves removing auxiliary verbs like "be" and replacing them with caps. (Figure 3.4).
- *Coordination rule*: the coordination rule simplifies the use of conjunctions like "and" by replacing them with a layer of interleaving spiders (Figure 3.5).

Another technique for reducing the width of the quantum circuit, prior to converting the string diagram into it, involves *removing cups* from the diagram by bending down the wires, as demonstrated in Figure 3.4. For instance, we replace the cup and the noun's state $I \to n$ with an effect $n^r \to I$ or $n^l \to I$, depending on which end of the cup the noun was. By doing so, we reduce the number of post-selections, which makes the model computationally more efficient – this becomes clearer during the subsequent phase.

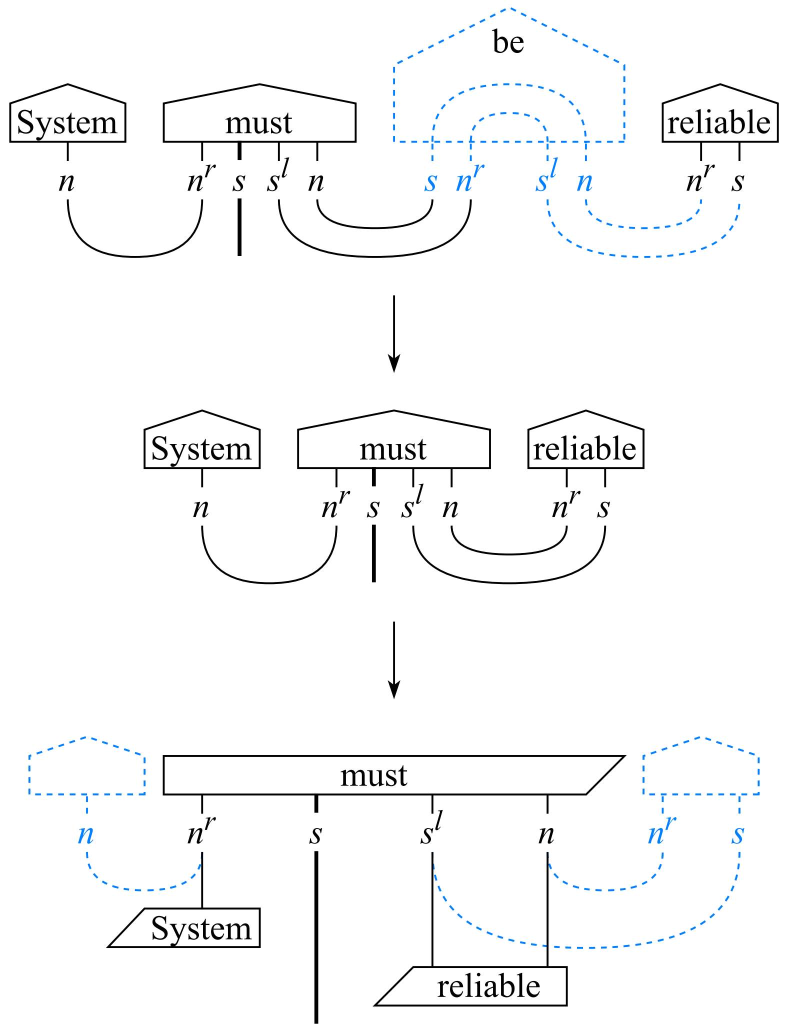


**Figure 3.4:** Auxiliary rewriting rule and cups removal.

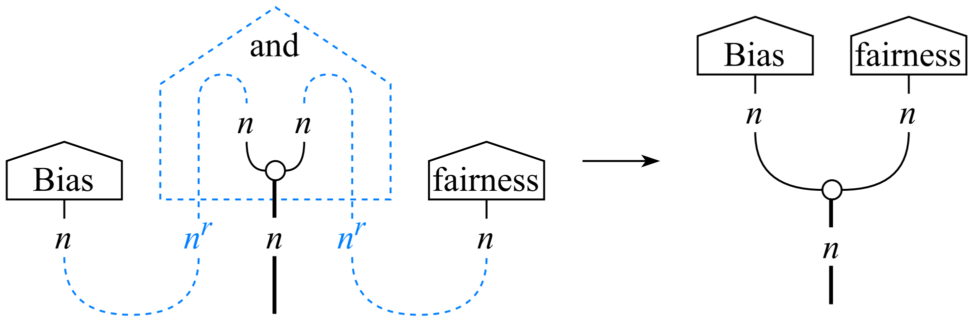


**Figure 3.5:** Coordination rewriting rule.

### 3.4.3 Parameterization

Until this stage of the pipeline, a sentence remains represented as an abstract string diagram, free from any low-level specifications like tensor dimensions or quantum gate selections. This abstract representation can be transformed into a tangible quantum circuit or tensor network by applying *ansätze*.

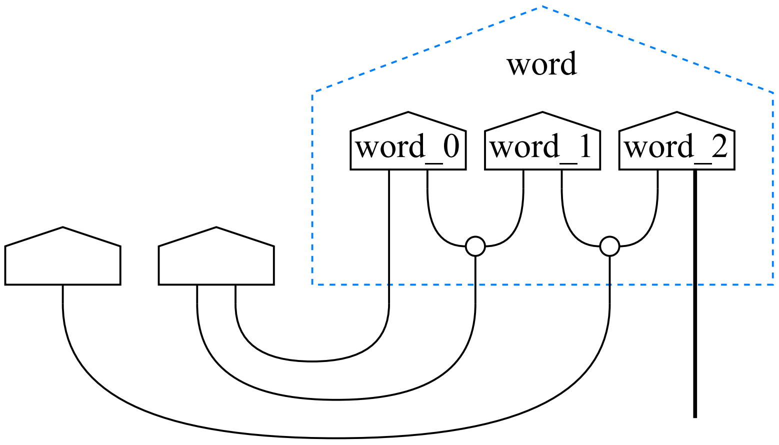


**Figure 3.6:** Split of a word into 3 lower order tensors using SpiderAnsatz.

In the classical implementation, the ansatz involves determining the values of $d_n$ and $d_s$, representing respectively the dimensions of the noun space and sentence space, to construct a concrete tensor network. To prevent the tensors associated with particular words – such as conjunctions – from becoming excessively large, lambeq incorporates ansätze for transforming tensors into different types of Matrix Product States (MPSs). The goal is to achieve the factorization of a large tensor into a product of smaller tensors arranged in a chain-like fashion. Specifically, as shown in Figure 3.6, the *SpiderAnsatz* enables us to decompose tensors with an order exceeding a specified threshold into lower-order tensors connected by spiders.

Regarding the quantum aspect, an ansatz plays a crucial role by providing a mapping that specifies essential elements, such as the quantities of qubits, denoted as $q_n$ and $q_s$, associated respectively with each wire of type "$n$" and "$s$," as well as the corresponding parameterized quantum states for each word. In our work, we won't delve extensively into the details of quantum ansätze specification, nevertheless, we will provide an explanation of the *instantaneous quantum polynomial* (IQP) ansatz which we have utilized in our experiments.

When using the *IQPAnsatz*, a word-state with only one output wire becomes a one-qubit-state prepared from $|0\rangle$ with a composition of parameterized rotation gates $R_x(\theta_1) \circ R_z(\theta_2) \circ R_x(\theta_3)$ defining an *Euler decomposition*. Word-states with more than one output wires become a $m$ qubits state consisting of all $m$ qubits prepared from $|0\rangle$ and followed by $d$-many IQP-layer [33]. Each IQP-layer is composed by an Hadamard gate on every qubit composed with $m-1$ $CR_z$ gates (a controlled version of $R_z$), connecting every neighbouring pair of qubits wires, and a final row of Hadamards. As a final point, cups of pregroup type $b$ are mapped to $q_b$-many nested Bell's effects, i.e., Bell's state conjugate transpose, which is implemented as a CNOT followed by a H gate on the control qubit and post-selection on $\langle 00|$.

We now have a comprehensive understanding of the rationale behind the costliness of cups: each cup implies $2q_b$ qubits that necessitate post-selection, hence, assuming that all potential outcomes are uniformly distributed, even the simple act of post-selecting 4 qubits (2 cups) to yield the outcome 0 results in only one out of $2^4$ shots meeting this condition, while discarding the rest. Moreover, this situation can become substantially worse when considering that the probability of observing the $p$ qubits yielding outcome 0 can be significantly lower than $1/(2^p)$.

Figure 3.7 illustrates the IQPAnsatz applied to the string diagram shown in Figure 3.4 before the removal of cups. Additionally, for the sake of comprehensiveness, we have also included the quantum circuit in Figure 3.8 corresponding to the string diagram after removing cups. It's noteworthy to observe that the components of the latter undergo transpositions, resulting in the arrangement of layers and rotations in the reverse order. In these examples we choose $q_n = 1$, $q_s = 1$ and $d = 1$ (IQP-layers).

For binary classification tasks we used $q_s = 1$ as we need 1 qubit outcome to make a binary classification, conversely, for multi-class classification we utilized $q_s = 2$ as we need at least 2 qubit to classify the four different classes.

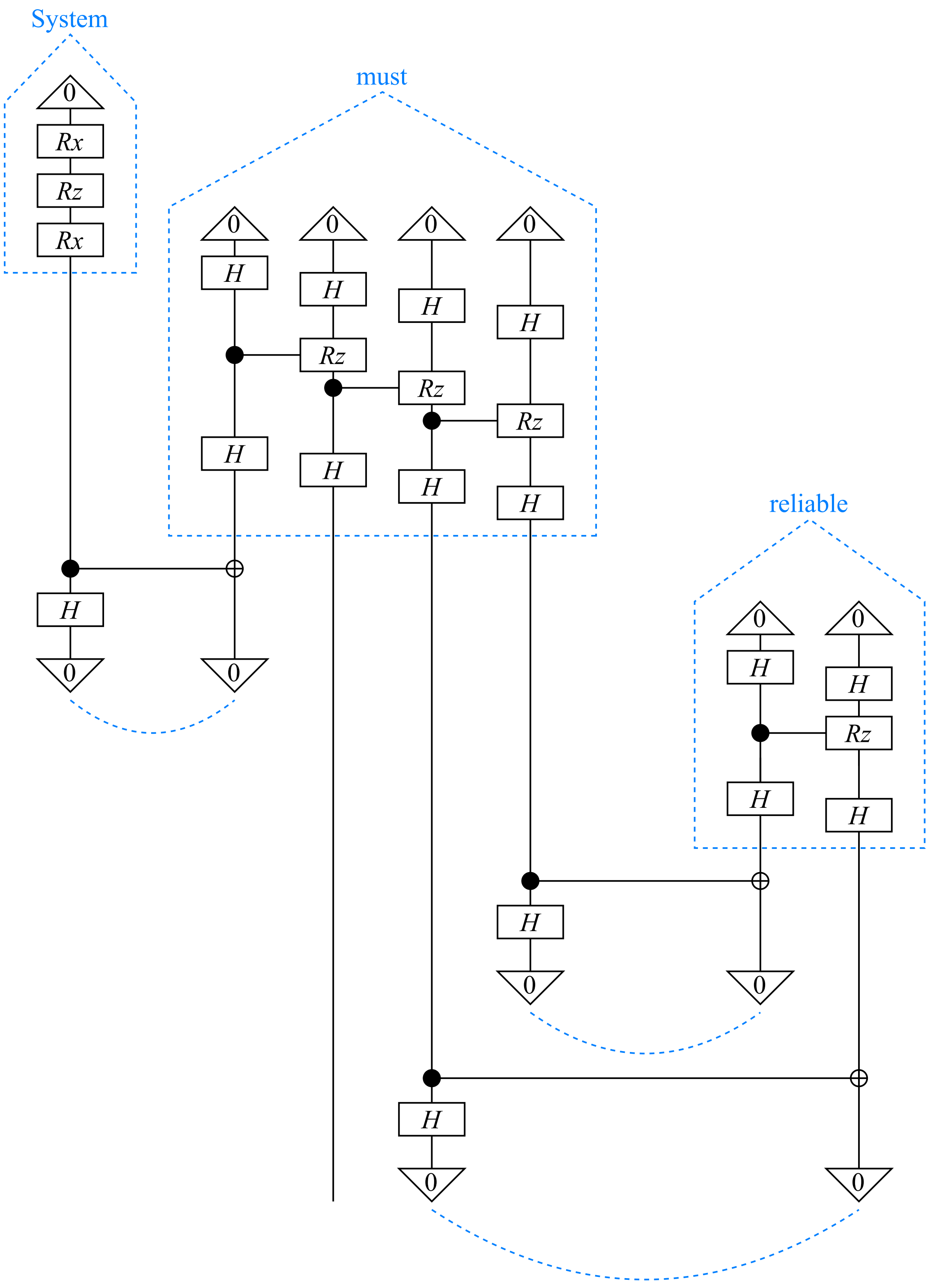


**Figure 3.7:** IQP circuit of 3.4 before cups removal.

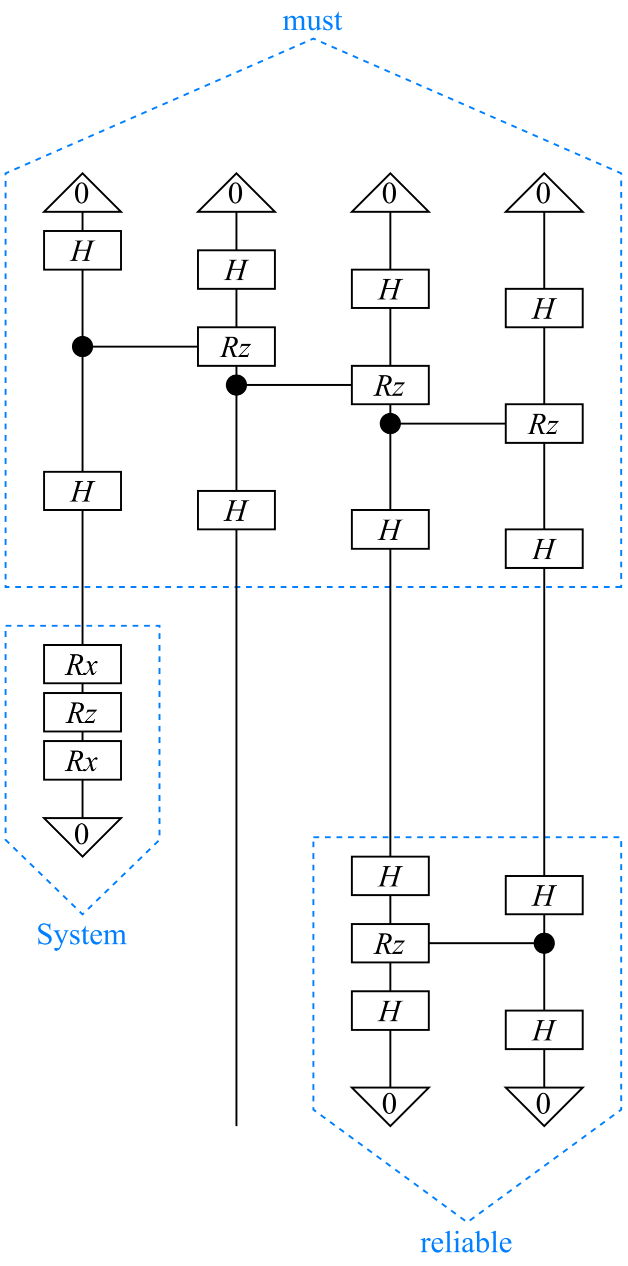


**Figure 3.8:** IQP circuit of 3.4 after cups removal.

### 3.4.4 Optimization

Machine learning truly comes into play when it comes to optimization, the goal is to fine-tune the model's parameters, denoted as a vector $\theta$, to minimize a given *loss function* $L(\theta)$. In the classical implementation, the optimizer modifies the tensors associated with words to generate sentence tensors that minimize the error produced by the classifier. In the quantum counterpart, there are no tensors; instead, we adjust the parameters of the rotational gates to yield the correct outcome in the circuit's output. The most popular optimization strategy is *gradient descent* (GD), which involves moving in the opposite direction of the loss function's gradient, i.e., the partial derivatives with respect to each parameter. The core formula for gradient descent involves updating the parameters iteratively as follows:

$$\theta_{t+1} = \theta_t - \alpha \nabla L(\theta_t) \ ,$$

where $\theta_t$ represents the parameter vector at iteration $t$, $\alpha$ is the *learning rate* determining the step size, and $\nabla L(\theta_t)$ denotes the gradient of the objective function with respect to the parameters. The learning rate $\alpha$ has a significant impact on the speed of learning: if $\alpha$ is too high, the algorithm may diverge, while if $\alpha$ is too low, the convergence of the algorithm is slowed down. A common practice is to make $\alpha_t$ a decreasing function of the number of iterations, $t$.

For the binary classification tasks, we use the *binary cross entropy* (BCE) as loss function which formula is given by:

$$BCE = -\frac{1}{N} \sum_{i=1}^{N} [y_i \cdot \log(p_i) + (1 - y_i) \cdot \log(1 - p_i)]$$

where $N$ is the number of samples, $y_i$ is the true label for the $i$-th sample, and $p_i$ is the predicted probability of the sample belonging to class 1.

Regarding multi-class classification, we use the *categorical cross entropy* (CCE), which is expressed as:

$$CCE = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} y_{i,j} \cdot \log(p_{i,j})$$

where $N$ is the number of samples, $C$ is the number of classes, $y_{i,j}$ is a binary indicator of whether class $j$ is the correct classification for sample $i$, and $p_{i,j}$ is the predicted probability of sample $i$ belonging to class $j$.

As optimization algorithm, we opted for the utilization of AdamW, an extension of the well-known Adam optimizer, which introduces a modified approach to *weight decay* regularization, addressing potential issues and enhancing the optimization process. Weight decay is a regularization technique used in machine learning to prevent overfitting during the training of a model, it involves adding a penalty term to the loss function, proportional to the squared magnitude of the weights. In this way, the model is encouraged to use smaller weights, helping to prevent complex models from fitting the training data too closely and improving the generalization performance on unseen data.

## 3.5 Validation Methods and Evaluation Criteria

In a ML pipeline, configuring hyperparameters, namely values that control various aspects of the algorithm and significantly influence its performance, is a crucial step for optimizing the learning process of each model. To identify pseudo-optimal hyperparameters, a commonly used technique is the *grid search*. This method involves defining a search space as a grid of hyperparameter values and systematically evaluating each position in the grid by testing all possible combinations of these values. After this process, combinations that yield the most balanced and effective results are selected. For our specific case, we employed grid search to fine-tune hyperparameters such as learning rate and batch size (the number of training examples utilized in one learning step). The snippet below illustrates how grid search was utilized, performing model evaluations among 8 different combinations of hyperparameters:

```python
for batch_size in [32, 64]:
    for learning_rate in [7e-3, 1e-2, 3e-2, 5e-2]:
        # perform model evaluation
```

To evaluate the accuracy and effectiveness of a machine learning model, one can conduct one or more assessments on the errors observed in the predictions. However, this evaluation solely offers insights into the model's performance on the training data, raising the possibility of the model being either underfitting or overfitting to the data. Consequently, it lacks an indication of how well the learning model will generalize to an independent or unseen dataset it has not encountered before. To address this concern, we implement *k-fold cross-validation* by partitioning the original dataset *k* times, with *k* set to 10 in our case, training the model on different subsets while using the remaining fold for validation. As our dataset is quite unbalanced, we apply a *stratified* version of cross-validation in order to ensure that the distribution of classes remains consistent across all folds. This prevents the occurrence of situations where a particular fold might have a disproportionate number of instances from one class, which could lead to biased model evaluations. By employing cross-validation and thereby reducing the impact of data variability and overfitting, we aim to obtain more reliable evaluation metrics for assessing the model's performance.

To assess the predictive effectiveness, in our experiments we utilized precision, recall, accuracy and F1-score to quantitatively measure the effectiveness of the model and address research questions. These evaluation metrics are outlined as the following:

- *Precision* ($P$): The ratio of correctly predicted positive observations to the total predicted positive observations. The formula is :

  $$P = \frac{TP}{TP + FP}$$

  where $TP$ is the number of true positives (correctly predicted positive observations) and $FP$ is the number of false positives.

- *Recall* ($R$): The ratio of correctly predicted positive observations to all observations. The formula is :

  $$R = \frac{TP}{TP + FN}$$

  where $FN$ is the number of false negatives.

- *Accuracy* ($ACC$): The ratio of the total number of correct predictions to the total number of observations, given by the formula:

  $$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

  where $TN$ is the number of true negatives.

- *F1-Score*: The weighted average of precision and recall, given by the formula:

$$F1 = 2 \cdot \frac{P \cdot R}{P + R}$$

  The F1-score is particularly useful in scenarios with imbalanced datasets, where the class of interest is underrepresented. As such, in our case it provides a balanced measure by considering both false positives and false negatives, making it a reliable metric for model performance in such situations.

## 3.6 Technological Frameworks and Experimental Setup

During this project, various frameworks and libraries were used to facilitate many of the operations. As mentioned earlier, Lambeq was the main framework that accompanied the entire QNLP pipeline – its backend, DisCoPy[3], allows computation with monoidal categories and manipulation of string diagrams. During the text preprocessing phase, the nltk[4] toolkit was used, providing a set of libraries for text processing tasks such as tokenization, stemming, and additionally, it provides corpora and lexical resources. In the optimization stage, and in general concerning ML tasks, we leverage the functionalities of PyTorch [5], a widely embraced and adaptable machine learning framework with seamless integration with Lambeq which stands out as the straightforward choice for handling tensor networks in classical experiments. In our quantum implementation, we employed PennyLane [6], a specialized quantum machine learning library which provides a versatile way to design, simulate, and optimize quantum circuits.

The experiments are conducted on the Google Colab platform[7] using virtual machines with increased RAM and no hardware acceleration for both classical and quantum experiments. It includes all the necessary tools, libraries, and configurations required for the process, also providing a way for researchers to replicate our experiments and validate our findings.

[3]discopy.org
[4]nltk.org
[5]pytorch.org
[6]pennylane.ai
[7]colab.research.google.com

CHAPTER 4

# Results Analysis

In this chapter, we present the key findings and conduct a thorough analysis to address our research questions. First, we outline the outcomes of the grid search, subsequently, using the hyperparameters that yielded the most effective model, we address the research questions by reporting the outcomes of the 10-fold cross-validation average values.

## 4.1 Grid Search Results

As shown in Chapter 3, through a grid search involving 8 combinations of hyperparameters, we fine-tuned the model. We executed this process for each task and implementation, the most meaningful results are explained in Table 4.2. All subsequent executions are performed using the following hyperparameters:

- *Binary classical DisCoCat*: learning rate = 3e-2, batch size = 64.
- *Multi-class classical DisCoCat*: learning rate = 1e-2, batch size = 64.
- *Binary quantum DisCoCat*: learning rate = 3e-2, batch size = 32.
- *Multi-class quantum DisCoCat*: learning rate = 5e-2, batch size = 64.

**Table 4.1:** Results of grid-search for each model.

| Hyperparameters | Precision | Recall | Accuracy | F1-Score |
|---|---|---|---|---|
| **Binary classical DisCoCat** | | | | |
| $lr$ = 3e-2, $bs$ = 64 | **0.86** | 0.79 | **0.85** | **0.83** |
| $lr$ = 1e-2, $bs$ = 32 | 0.84 | **0.80** | 0.80 | 0.81 |
| $lr$ = 3e-2, $bs$ = 32 | **0.86** | 0.79 | 0.81 | **0.83** |
| **Multi-class classical DisCoCat** | | | | |
| $lr$ = 1e-2, $bs$ = 32 | 0.62 | 0.62 | **0.64** | 0.62 |
| $lr$ = 1e-2, $bs$ = 64 | **0.65** | **0.64** | **0.64** | **0.65** |
| $lr$ = 3e-2, $bs$ = 64 | 0.62 | **0.64** | 0.62 | 0.64 |
| **Binary quantum DisCoCat** | | | | |
| $lr$ = 3e-2, $bs$ = 64 | 0.92 | **0.87** | 0.88 | 0.89 |
| $lr$ = 3e-2, $bs$ = 32 | **0.96** | **0.87** | **0.93** | **0.95** |
| **Multi-class quantum DisCoCat** | | | | |
| $lr$ = 5e-2, $bs$ = 64 | **0.78** | 0.68 | **0.76** | **0.73** |
| $lr$ = 3e-2, $bs$ = 64 | 0.75 | **0.70** | 0.70 | 0.71 |

## 4.2 Research Question 1

**RQ$_1$.** *To what extent can shallow ML algorithms classify the four most frequent NFR types in the PROMISE dataset: usability, security, operational, and performance?*

Firstly, let's present the results of experiments related to the utilization of word-embeddings using Bag-of-Words, TF-IDF, and Word2Vec for the task of NFRs binary and multi-class classification. For BoW and TF-IDF, 100-dimensional embeddings were employed. As for Word2Vec, we utilized "word2vec-google-news-300," representing pre-trained vectors trained on a subset of the Google News dataset, encompassing around 100 billion words – this particular model contains 300-dimensional vectors. Table 4.2 shows the best four shallow ML models employed for the binary and the multi-class classification task, with the bold highlighted results indicating the best outcome for the respective vectorization technique. The outcomes aligns with expectations, given Word2Vec's ability to capture more nuanced semantic relationships between words by leveraging dense representations and contextual information, while the results obtained with BoW and TF-IDF are generally poor.

**Table 4.2:** Results of accuracy achieved with the best shallow ML models.

| **Binary classification** | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | **Word2Vec** | | | | **TF-IDF** | | | | **Bag-of-Words** | | | |
| | Precision | Recall | Accuracy | F1 | Precision | Recall | Accuracy | F1 | Precision | Recall | Accuracy | F1 |
| **Security (SE)** | | | | | | | | | | | | |
| CategoricalNB | 0.94 | 0.92 | 0.94 | 0.94 | 0.84 | 0.79 | 0.81 | 0.81 | 0.76 | 0.70 | 0.74 | 0.75 |
| ExtraTreesClassifier | **0.96** | **0.95** | **0.95** | **0.95** | 0.83 | 0.83 | **0.83** | **0.83** | 0.74 | **0.78** | **0.77** | **0.76** |
| BernoulliNB | 0.92 | 0.93 | 0.91 | 0.92 | **0.87** | 0.78 | 0.82 | **0.83** | **0.77** | 0.74 | 0.74 | 0.74 |
| NearestCentroid | 0.93 | 0.89 | 0.92 | 0.92 | 0.80 | **0.84** | 0.81 | 0.80 | 0.75 | 0.74 | 0.74 | 0.75 |
| **Usability (US)** | | | | | | | | | | | | |
| CategoricalNB | **0.92** | **0.89** | **0.90** | **0.91** | **0.75** | **0.70** | **0.70** | **0.71** | 0.68 | 0.65 | **0.68** | **0.68** |
| ExtraTreesClassifier | 0.90 | 0.88 | **0.90** | 0.90 | 0.70 | 0.64 | 0.69 | 0.69 | **0.70** | 0.62 | **0.68** | **0.68** |
| BernoulliNB | 0.88 | 0.83 | 0.88 | 0.87 | 0.72 | 0.65 | 0.67 | 0.68 | 0.63 | **0.69** | 0.65 | **0.68** |
| NearestCentroid | 0.89 | 0.88 | 0.88 | 0.88 | 0.70 | 0.65 | 0.67 | 0.68 | 0.65 | 0.62 | 0.64 | 0.64 |
| **Operational (O)** | | | | | | | | | | | | |
| CategoricalNB | **0.98** | **0.95** | **0.95** | **0.95** | 0.84 | 0.80 | 0.81 | 0.82 | **0.80** | **0.77** | 0.76 | **0.77** |
| ExtraTreesClassifier | 0.90 | 0.85 | 0.87 | 0.87 | **0.86** | **0.83** | **0.83** | **0.84** | **0.80** | 0.75 | **0.77** | **0.77** |
| BernoulliNB | 0.93 | 0.90 | 0.92 | 0.91 | 0.84 | 0.78 | 0.81 | 0.79 | 0.78 | 0.75 | **0.77** | **0.77** |
| NearestCentroid | 0.92 | 0.88 | 0.91 | 0.91 | 0.84 | 0.78 | 0.79 | 0.79 | 0.78 | **0.77** | **0.77** | **0.77** |
| **Performance (PE)** | | | | | | | | | | | | |
| CategoricalNB | **0.92** | **0.89** | **0.91** | **0.92** | 0.79 | 0..79 | 0.81 | 0.79 | **0.76** | **0.75** | **0.74** | **0.75** |
| ExtraTreesClassifier | 0.90 | 0.86 | 0.88 | 0.87 | **0.84** | 0.80 | **0.84** | **0.84** | 0.73 | 0.70 | 0.73 | 0.71 |
| BernoulliNB | 0.85 | 0.80 | 0.87 | 0.85 | 0.80 | **0.82** | 0.79 | 0.80 | 0.74 | **0.75** | **0.74** | 0.74 |
| NearestCentroid | 0.86 | 0.82 | 0.87 | 0.85 | 0.78 | 0.77 | 0.79 | 0.77 | 0.75 | 0.74 | 0.72 | **0.75** |
| **Multi-class classification** | | | | | | | | | | | | |
| CategoricalNB | 0.75 | **0.75** | 0.74 | **0.75** | **0.70** | 0.64 | 0.67 | 0.68 | 0.64 | 0.59 | 0.63 | 0.63 |
| ExtraTreesClassifier | **0.78** | 0.74 | 0.74 | 0.74 | **0.70** | **0.65** | **0.68** | **0.68** | 0.61 | 0.61 | 0.63 | 0.61 |
| BernoulliNB | 0.72 | 0.73 | 0.72 | 0.72 | **0.70** | 0.63 | 0.67 | 0.66 | 0.61 | **0.62** | 0.61 | 0.61 |
| NearestCentroid | 0.75 | 0.76 | **0.75** | **0.75** | **0.70** | 0.63 | 0.66 | 0.66 | **0.65** | 0.58 | **0.65** | **0.65** |

**Answer to RQ$_1$.** In summary of the results pertaining to the first research question, vectorization techniques based solely on word frequency do not ensure high accuracy in the binary and multi-class classification of the four most frequent subcategories of NFRs in the manually filtered PROMISE dataset. On the other hand, Word2Vec proves to achieve excellent results, with Security (SE) and Operational (O) being the most accurately classified subcategories.

## 4.3 Research Question 2

**RQ$_2$.** *Is the classical implementation of DisCoCat accurate at least as the best shallow ML algorithms in classifying the four most frequent NFR subcategories in the PROMISE dataset: usability, security, operational, and performance?*

Using the classical implementation of DisCoCat, which translates requirements into tensor networks, experiments were conducted with different dimensions for the sentence ($d_s$) and noun ($d_n$) spaces. The best results were obtained with the configuration $d_n = 2$, $d_s = 2$, indicating that using a larger space affects the model's complexity and, consequently, its ability to generalize to the validation set, considering our small dataset. Additionally, it impacts the computational complexity of the model since we contract tensors in higher dimensional spaces.

The classical DisCoCat achieved better results in the binary classification of security and operational requirements, reaching approximately 84% precision and 80% recall for both, similar to the best shallow ML model involving TF-IDF vectorization technique. Overall, it demonstrates comparability with TF-IDF results, despite the significant reduction in tensor dimensions (2 compared to the 100 employed in TF-IDF and BoW embeddings), and significantly better results compared to BoW technique. Nonetheless, the results fall considerably below the performance achieved with Word2vec for both binary and multi-class classification.

**Answer to RQ$_2$.** The results obtained from the classical implementation of DisCoCat showed inferior classification capabilities compared to Word2vec and comparable to TF-IDF, despite using tensors of much smaller dimensions. Ultimately, the results of classical DisCoCat are promising, comparable to term frequency-based vectorization techniques, but it does not achieve the effectiveness in classifying NFRs guaranteed by shallow ML algorithms employing vectorization techniques such as Word2vec.

**Table 4.3:** Results achieved with classical DisCoCat.

| **Binary classification** | | | | |
|---|---|---|---|---|
| **Tensor space dim.** | **Precision** | **Recall** | **Accuracy** | **F1-Score** |
| | | **Security (SE)** | | |
| $d_n = 2, d_s = 2$ | **0.86** | **0.80** | **0.85** | **0.84** |
| $d_n = 2, d_s = 4$ | 0.80 | **0.78** | 0.80 | 0.81 |
| | | **Usability (US)** | | |
| $d_n = 2, d_s = 2$ | **0.78** | **0.76** | **0.76** | **0.78** |
| $d_n = 2, d_s = 4$ | **0.78** | 0.74 | **0.76** | **0.78** |
| | | **Operational (O)** | | |
| $d_n = 2, d_s = 2$ | **0.84** | **0.80** | **0.82** | **0.84** |
| $d_n = 2, d_s = 4$ | 0.79 | **0.78** | 0.75 | 0.79 |
| | | **Performance (PE)** | | |
| $d_n = 2, d_s = 2$ | **0.81** | **0.79** | **0.81** | **0.80** |
| $d_n = 2, d_s = 4$ | 0.80 | 0.78 | 0.80 | 0.81 |
| **Multi-class classification** | | | | |
| $d_n = 2, d_s = 2$ | **0.66** | **0.62** | **0.65** | **0.64** |
| $d_n = 2, d_s = 4$ | 0.62 | 0.58 | 0.60 | 0.62 |

## 4.4 Research Question 3

🔍 **RQ$_3$.** *Is the quantum implementation of DisCoCat accurate at least as the best shallow ML algorithms in classifying the four most frequent NFR subcategories in the PROMISE dataset: usability, security, operational, and performance?*

Similarly to the QNLP pipeline for classical experiments, at the end of the rewriting phase, the string diagrams were parameterized into quantum circuits using the IQPAnsatz. The results in Table 4.4 display multiple executions using various hyperparameter configurations, specifically focusing on the number of qubits for the name ($q_n$) and sentence ($q_s$) space, as well as the number of IQP-layers ($d$) in the circuits. As observed, both accuracy and F1-score values are significantly superior compared to the classical DisCoCat counterpart. In many cases, quantum DisCoCat achieves comparable or even superior performance against the results yielded by Word2Vec, despite having significantly fewer model parameters.

Due to imposed resource constraints, we were unable to further increase the hyperparameter $d$ beyond the value of 5. In the case of binary classification, excessively high values of $d$ negatively impact the results, generating a model that is too complex for the given task. This is in contrast to the multi-class scenario, where a higher number of IQP-layers helps to discriminate more effectively among the four classes, suggesting that with additional resources, even better results could be achieved.

In Figure 4.1, the multi-class learning curves are presented, highlighting the mean value across various folds of cross-validation, while the shaded area represents the maximum and minimum values among all folds for each epoch. The substantial spread in the validation set concerning the training set illustrates the significant influence of data splitting on the results and model performance. This is attributed to the small size of the dataset and the diversity of words; for instance, some infrequent words may only appear in the validation set, introducing noise and making it challenging to effectively learn their meanings.

**Table 4.4:** Results achieved with quantum DisCoCat.

| Tensor space dim. | Precision | Recall | Accuracy | F1-Score |
|---|---|---|---|---|
| **Binary classification** | | | | |
| **Security (SE)** | | | | |
| $q_n = 1, q_s = 1, d = 1$ | **0.98** | **0.89** | **0.93** | **0.94** |
| $q_n = 1, q_s = 1, d = 3$ | 0.93 | **0.89** | 0.91 | 0.92 |
| $q_n = 1, q_s = 1, d = 5$ | 0.90 | 0.86 | **0.93** | 0.88 |
| **Usability (US)** | | | | |
| $q_n = 1, q_s = 1, d = 1$ | **0.95** | **0.85** | **0.88** | **0.91** |
| $q_n = 1, q_s = 1, d = 3$ | 0.90 | **0.85** | **0.88** | 0.90 |
| $q_n = 1, q_s = 1, d = 5$ | 0.89 | **0.85** | 0.87 | 0.88 |
| **Operational (O)** | | | | |
| $q_n = 1, q_s = 1, d = 1$ | **0.97** | **0.90** | **0.94** | **0.96** |
| $q_n = 1, q_s = 1, d = 3$ | 0.93 | 0.89 | **0.94** | 0.91 |
| $q_n = 1, q_s = 1, d = 5$ | 0.93 | 0.85 | 0.88 | 0.91 |
| **Performance (PE)** | | | | |
| $q_n = 1, q_s = 1, d = 1$ | **0.90** | **0.90** | 0.88 | **0.90** |
| $q_n = 1, q_s = 1, d = 3$ | 0.85 | 0.87 | 0.83 | 0.85 |
| $q_n = 1, q_s = 1, d = 5$ | 0.86 | 0.84 | **0.89** | 0.86 |
| **Multi-class classification** | | | | |
| $q_n = 1, q_s = 2, d = 1$ | 0.70 | 0.68 | 0.68 | 0.68 |
| $q_n = 1, q_s = 2, d = 3$ | 0.74 | **0.71** | 0.70 | 0.72 |
| $q_n = 1, q_s = 2, d = 5$ | **0.80** | **0.71** | **0.76** | **0.74** |

In Figure 4.2, we present the binary classification, showing different lines on the graph for various subcategories of NFRs. As we can observe, operational requirements are the easiest to classify, which can be attributed to the extensive use of words related to technologies, architectures, and programming languages that appear predominantly in operational requirements. Security requirements are also easy to classify – the grammatical structure often involves negating actions for a specific type of user, and also in this case security domain-specific words are frequently used. Performance and usability requirements remain the most challenging to classify, possibly due to many of these requirements having a similar grammatical structure. Taking, for example, the following two requirements, one related to usability and the other to performance:

*US: Users are able to successfully reserve rooms in 5 minutes*

*PE: Add new products to website takes 2 minutes*

It is clear that the difference is challenging even for manual classification, where the first one pertains to user ease in *"reserving rooms in 5 minutes,"* while the second establishes an upper time limit for an *"add new product"* operation.

☞ **Answer to RQ3.** In both binary and multi-class NFRs classification tasks, quantum DisCoCat significantly surpassed the performance of BoW and TF-IDF, and it achieved comparable or even superior accuracy compared to the Word2Vec text vectorization technique, highlighting its potential for a more effective and nuanced representation of words and sentences. Ultimately, the quantum DisCoCat is accurate at least as the best shallow ML even when considering Word2Vec as text vectorization technique.

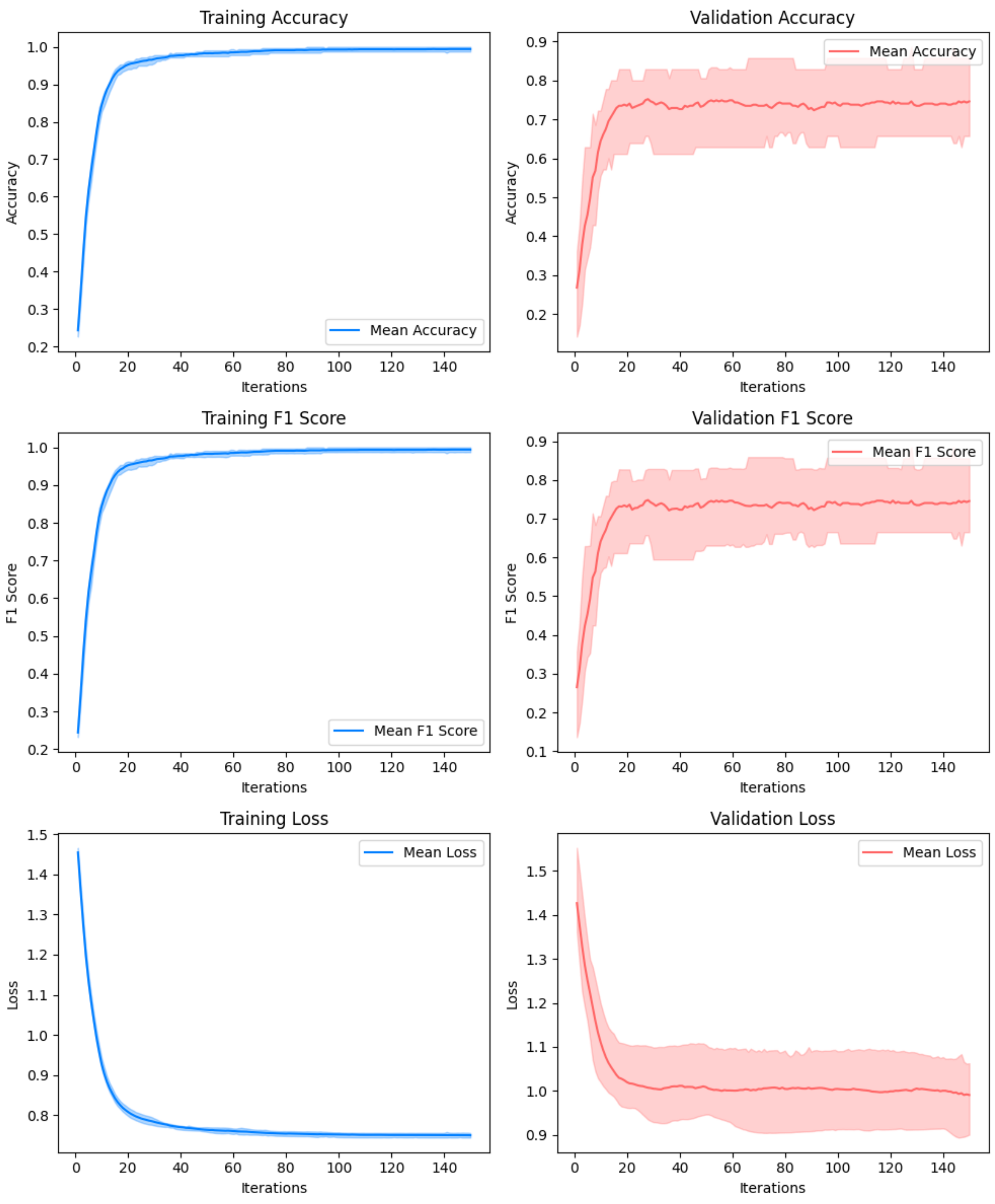


**Figure 4.1:** Learning curves multi-class classification with quantum DisCoCat.

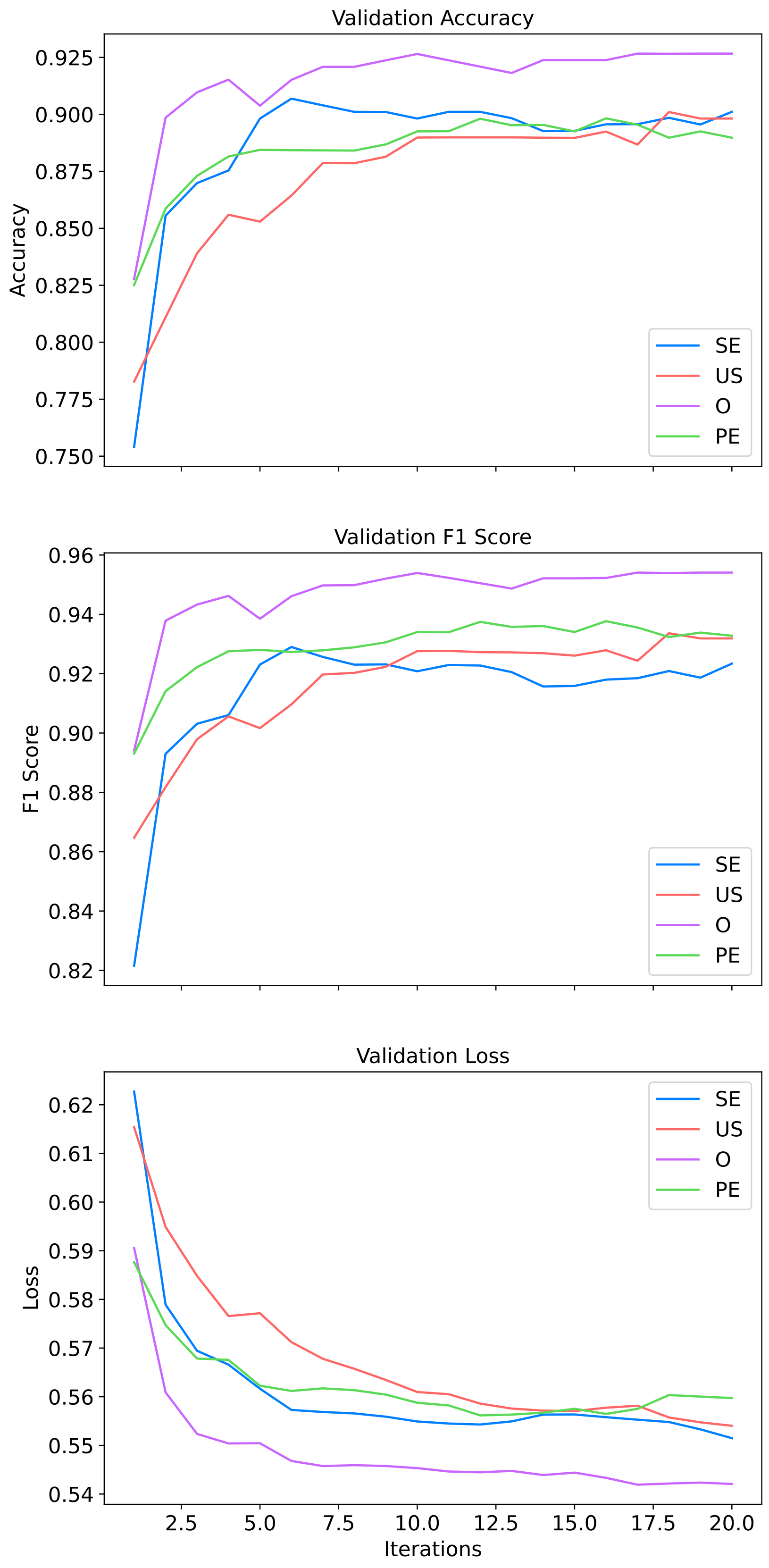


**Figure 4.2:** NFRs subcategories binary classification with quantum DisCoCat.

## 4.5 Research Question 4

**RQ$_4$.** *To what extent does the grammatical structure of sentences impact NFRs classification when following the quantum approach?*

During the parsing phase, we employed alternative linear readers that disregard grammar and concentrate on a single pregroup symbol. Consequently, every word is assigned to the same Hilbert space. Table 4.5 displays the results of experiments with three linear readers presented in Chapter 3, using $q_s = 1$ in the binary case and $q_s = 2$ in the multi-class case. Despite the simplified grammatical structure, both manually in the dataset filtering phase and automatically in the string diagram rewriting process, the results produced by DisCoCat consistently outperformed or matched various linear readers meaning that grammar provides an additional value in this specific scenario.

**Answer to RQ$_4$.** The grammatical structure's impact on NFRs classification within the quantum approach is noteworthy. Despite substantial simplifications in the grammatical structure, grammar provides useful insights on the meaning of NFRs description as it allows the quantum approach to capture nuanced relationships between words, providing a level of depth and understanding that a traditional bag-of-words representation might overlook.

**Table 4.5:** Results achieved with quantum linear readers.

| **Tensor space dim.** | **Precision** | **Recall** | **Accuracy** | **F1-Score** |
|---|---|---|---|---|
| **Binary classification** | | | | |
| **Security (SE)** | | | | |
| SpidersReader | 0.94 | **0.89** | 0.89 | **0.94** |
| CupsReader | 0.90 | 0.84 | 0.85 | 0.89 |
| StairsReader | 0.90 | 0.82 | 0.89 | 0.89 |
| **DisCoCat** | **0.98** | 0.89 | **0.93** | **0.94** |
| **Usability (US)** | | | | |
| SpidersReader | 0.91 | 0.82 | **0.92** | 0.90 |
| CupsReader | 0.86 | **0.86** | 0.89 | 0.86 |
| StairsReader | 0.93 | **0.86** | 0.89 | 0.90 |
| **DisCoCat** | **0.95** | 0.85 | **0.88** | **0.91** |
| **Operational (O)** | | | | |
| SpidersReader | 0.93 | 0.88 | 0.91 | 0.92 |
| CupsReader | 0.91 | **0.90** | 0.91 | 0.91 |
| StairsReader | 0.94 | 0.87 | **0.93** | 0.92 |
| **DisCoCat** | **0.97** | **0.90** | **0.93** | **0.96** |
| **Performance (PE)** | | | | |
| SpidersReader | **0.90** | 0.85 | **0.89** | 0.88 |
| CupsReader | 0.89 | 0.83 | 0.85 | 0.87 |
| StairsReader | 0.89 | 0.87 | **0.89** | 0.88 |
| **DisCoCat** | **0.90** | 0.90 | 0.88 | **0.90** |
| **Multi-class classification** | | | | |
| SpidersReader | 0.78 | 0.70 | 0.72 | 0.73 |
| CupsReader | 0.75 | **0.73** | 0.70 | **0.74** |
| StairsReader | 0.78 | 0.70 | 0.73 | **0.74** |
| **DisCoCat** | **0.80** | 0.71 | **0.76** | **0.74** |

CHAPTER 5

# Threats To Validity

This chapter discusses potential threats to the validity of our study, addressing factors that could impact the generalizability and the overall credibility of our findings. Ensuring the robustness and reliability of our study is paramount to drawing meaningful conclusions and establish the groundwork for future developments in the appropriate direction.

## 5.1 Internal Validity

Internal validity concerns the accuracy of causal inferences within our study that might have influenced the relationship between treatment and outcome.

#### Dataset simplification

A crucial consideration in our study revolve around the manual filtering of the NFRs sentences of the PROMISE dataset, undertaken to facilitate the translation process and stay within the limits imposed by resources. However, this simplification could potentially impact internal validity as it is acknowledged that more intricate quantum circuits might yield different results, and the trade-off between circuit complexity and performance should be carefully considered.

#### Quantum circuit design

The specific design and configuration of our quantum circuits for NFRs classification introduce an element of internal validity consideration. Variations in quantum circuit architectures, gate configurations, or other hyperparameters may influence the performance of the quantum model. However limited time and extremely long simulation run impose a crucial constraint so future investigations could explore different quantum circuit designs to assess the robustness and consistency of our findings.

## 5.2 External Validity

External validity addresses the generalizability of our study beyond the specific conditions and dataset used.

#### Dataset Specificity

Our study focused specifically on the PROMISE NFR dataset, which may limit the external validity to this particular dataset. It is recognized that results may vary when applied to different NLP tasks or datasets. Future research endeavors could explore a broader range of datasets to enhance the external validity of our NFRs classification findings.

#### Model Applicability

The applicability of our quantum and classical models may be context-dependent. While they demonstrated effectiveness for NFRs classification, their performance might differ when applied to diverse domains or tasks. Evaluating the models on various real-world applications and domains could provide a more comprehensive understanding of their generalizability.

## 5.3 Conclusion Validity

Conclusion validity addresses the accuracy of the conclusions drawn from the data.

#### Shallow ML models selection

To ensure conclusion validity, it is imperative to explore the widest range of alternative shallow machine learning models as the ever-evolving landscape of machine learning presents a myriad of algorithms, each with its strengths and weaknesses.

#### Evaluation Metrics

Our choice of evaluation metrics, including accuracy and F1-score, may not capture all nuances of NLP tasks. To enhance conclusion validity, future investigations could include additional metrics or explore task-specific measures that provide a more comprehensive assessment of NFRs classification performance.

## 5.4 Mitigation Strategies

To address these potential threats, we took a meticulous approach during the manual filtering steps, ensuring that sentence modifications aligned with the underlying concept. Specifically, we altered infrequent words only when they were non-characteristic of the respective sentences and removed phrases that did not contribute additional details to the requirements. Additionally, our study involved a comprehensive comparison of over 30 (considering only the top 4) different shallow machine learning models, employing rigorous cross-validation techniques.

It is important to note that while our study has provided valuable insights, there is still room for further exploration. Future endeavors could delve into more intricate quantum circuits, explore diverse datasets, and incorporate additional evaluation metrics. These efforts would contribute to a more comprehensive understanding and validation of our NFRs classification study.

CHAPTER 6

# Conclusion and Future Work

In the software development life cycle, the foremost phase is requirements engineering which holds crucial importance as it sets the foundation for the entire development process. *Non-functional requirements* (NFRs) play a vital role in defining the system's properties and constraints, their early identification significantly influences key decisions regarding system architecture, and failure to manage these requirements properly can lead to project failure. Automating this process through machine learning (ML) is necessary to reduce the tedious and time-consuming nature of manual extraction from documents, mitigating the susceptibility to various errors.

In this thesis, we delved into a quantum approach for addressing the real-world NLP challenge of NFRs classification using the DisCoCat framework, rooted in the field of computational linguistics. Our study highlights the effectiveness of the quantum model, showcasing promising overall results and providing valuable insights into the potential of quantum computing. By harnessing the compositional nature of language, classical implementation relying on tensor network has obtained comparable results to shallow ML employing frequency-based vectorization techniques – specifically BoW and TF-IDF – and its quantum counterpart outperformed TF-IDF achieving comparable results to a more powerful technique namely Word2Vec, all while significantly reducing the number of parameters.

Moreover, the model demonstrated slightly superior performance compared to other linear models which do not take into account the grammatical composition of sentences, despite still computing a tensor of the sentences. This suggests that, even though sentences were simplified for computational reasons, grammar remains a valuable aspect in this scenario, enhancing classification accuracy.

The findings of this study open up several avenues for future research. First, further exploration can be done to enhance the performance of the linear readers by incorporating additional linguistic features or exploring different parsing techniques. Additionally, investigating the impact of different preprocessing strategies on the performance of the readers can provide insights into the trade-off between grammatical structure and semantic information. Furthermore, as the field of quantum computing continues to advance, exploring the potential speedup that can be achieved on real run on quantum computers is an intriguing direction for future investigation.

Finally, the DisCoCat framework can be applied to other natural language processing tasks beyond requirement classification. Exploring its potential in sentiment analysis, text summarization, question-answering, or, eventually, extending this approach to other types of languages, e.g. programming languages, could prove to be effective since their fixed and limited grammar makes them well-suited for compositional modeling techniques like the one employed in our study.

# Bibliography

[1] I. Sommerville and P. Sawyer, *Requirements Engineering: A Good Practice Guide*, 1st ed. USA: John Wiley & Sons, Inc., 1997. (Citato alle pagine 1 e 6)

[2] W. Behutiye, P. Karhapää, D. Costal, M. Oivo, and X. Franch, *Non-functional Requirements Documentation in Agile Software Development: Challenges and Solution Proposal*. Springer International Publishing, 2017, p. 515–522. (Citato a pagina 1)

[3] V. Bajpai and R. Gorthi, "On non-functional requirements: A survey," in *2012 IEEE Students' Conference on Electrical, Electronics and Computer Science*, 2012, pp. 1–4. (Citato alle pagine 1 e 8)

[4] J. Eckhardt, A. Vogelsang, and D. M. Fernández, "Are "non-functional" requirements really non-functional? an investigation of non-functional requirements in practice," *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, pp. 832–842, 2016. (Citato alle pagine 2 e 6)

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017. (Citato a pagina 2)

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019. (Citato a pagina 2)

[7] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986. (Citato a pagina 2)

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 1997. (Citato a pagina 2)

[9] H. Zhao, H. Chen, F. Yang, N. Liu, H. Deng, H. Cai, S. Wang, D. Yin, and M. Du, "Explainability for large language models: A survey," 2023. (Citato a pagina 2)

[10] Z. S. Harris, "Distributional structure," *WORD*, vol. 10, no. 2-3, pp. 146–162, 1954. (Citato a pagina 2)

[11] B. Coecke, M. Sadrzadeh, and S. Clark, "Mathematical foundations for a compositional distributional model of meaning," 2010. (Citato alle pagine 2 e 18)

[12] K. Meichanetzidis, A. Toumi, G. de Felice, and B. Coecke, "Grammar-aware sentence classification on quantum computers," *Quantum Machine Intelligence*, vol. 5, 2023. (Citato alle pagine 3 e 21)

[13] R. Lorenz, A. Pearson, K. Meichanetzidis, D. Kartsaklis, and B. Coecke, "QNLP in practice: Running compositional models of meaning on a quantum computer," pp. 1305–1342, 2023. (Citato alle pagine 3 e 21)

[14] D. M. Fernández, S. Wagner, M. Kalinowski, M. Felderer, P. Mafra, A. Vetrò, T. Conte, M.-T. Christiansson, D. Greer, C. Lassenius, T. Männistö, M. Nayabi, M. Oivo, B. Penzenstadler, D. Pfahl, R. Prikladnicki, G. Ruhe, A. Schekelmann, S. Sen, R. Spinola, A. Tuzcu, J. L. de la Vara, and R. Wieringa, "Naming the pain in requirements engineering," *Empirical Software Engineering*, vol. 22, no. 5, pp. 2298–2338, oct 2016. (Citato a pagina 6)

[15] J. Sayyad Shirabad and T. Menzies, "The promise repository of software engineering databases." School of Information Technology and Engineering, University of Ottawa, Canada, 2005. [Online]. Available: http://promise.site.uottawa.ca/SERepository (Citato alle pagine 6, 20 e 24)

[16] R. R. Maiti and F. J. Mitropoulos, "Capturing, eliciting, predicting and prioritizing (cepp) non-functional requirements metadata during the early stages

of agile software development," in *SoutheastCon 2015*, 2015, pp. 1–8. (Citato a pagina 8)

[17] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, "A survey on bias and fairness in machine learning," vol. 54, 2021. (Citato a pagina 8)

[18] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, "The theory of variational hybrid quantum-classical algorithms," *New Journal of Physics*, vol. 18, p. 023023, 2016. (Citato a pagina 14)

[19] J. Spall, "Implementation of the simultaneous perturbation algorithm for stochastic optimization," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 34, pp. 817–823, 1998. (Citato a pagina 14)

[20] J. Lambek, "From word to sentence: A pregroup analysis of the object pronoun who(m)," *Journal of Logic, Language, and Information*, vol. 16, pp. 303–323, 2007. (Citato a pagina 17)

[21] W. Zeng and B. Coecke, "Quantum algorithms for compositional natural language processing," *Electronic Proceedings in Theoretical Computer Science*, vol. 221, pp. 67–75, 2016. (Citato alle pagine 19 e 21)

[22] M. Lu and P. Liang, "Automatic classification of non-functional requirements from augmented app user reviews," 06 2017. (Citato a pagina 20)

[23] F. Dalpiaz, D. Dell'Anna, F. B. Aydemir, and S. Çevikol, "Requirements classification with interpretable machine learning and dependency parsing," in *2019 IEEE 27th International Requirements Engineering Conference (RE)*, 2019, pp. 142–152. (Citato alle pagine 20 e 24)

[24] T. Hey, J. Keim, A. Koziolek, and W. F. Tichy, "Norbert: Transfer learning for requirements classification," in *2020 IEEE 28th International Requirements Engineering Conference (RE)*, 2020, pp. 169–179. (Citato a pagina 20)

[25] F. Casillo, V. Deufemia, and C. Gravino, "Detecting privacy requirements from user stories with nlp transfer learning models," *Information and Software Technology*, vol. 146, p. 106853, 2022. (Citato a pagina 20)

[26] B. Coecke, G. de Felice, K. Meichanetzidis, and A. Toumi, “Foundations for near-term quantum natural language processing,” 2020. (Citato a pagina 21)

[27] V. Martinez and G. Leroy-Meline, “A multiclass q-nlp sentiment analysis experiment using discocat,” 2022. (Citato a pagina 21)

[28] Z. Kurtanović and W. Maalej, “Automatically classifying functional and non-functional requirements using supervised machine learning,” in *2017 IEEE 25th International Requirements Engineering Conference (RE)*, 2017, pp. 490–495. (Citato a pagina 21)

[29] V. R. B. G. Caldiera and H. D. Rombach, “The goal question metric approach,” *Encyclopedia of software engineering*, pp. 528–532, 1994. (Citato a pagina 22)

[30] D. Kartsaklis, I. Fan, R. Yeung, A. Pearson, R. Lorenz, A. Toumi, G. de Felice, K. Meichanetzidis, S. Clark, and B. Coecke, “lambeq: An Efficient High-Level Python Library for Quantum NLP,” 2021. (Citato a pagina 28)

[31] R. Yeung and D. Kartsaklis, “A ccg-based version of the discocat framework,” 2021. (Citato a pagina 28)

[32] M. Steedman, *The Syntactic Process*. The MIT Press, 2001. (Citato a pagina 28)

[33] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, “Supervised learning with quantum-enhanced feature spaces,” *Nature*, vol. 567, pp. 209–212, 2019. (Citato a pagina 34)

# Acknowledgements

I would like to express my gratitude to Professor Fabio Palomba, my thesis advisor, for encouraging me to tackle a challenging topic, for his invaluable guidance throughout my academic journey, and for making every lesson not only an intellectual but also a personal growth experience. Additionally, I greatly appreciate the extreme professionalism and expertise of Dr. Francesco Casillo, whom I thank for his patience and availability.

I extend my thanks to my parents for providing me with the opportunity to pursue this academic journey, for encouraging me during tough times, and for teaching me that hard work always pays off. I want to express my gratitude to my brother, Giuseppe, who has been a source of inspiration and his guidance has been essential to me.

I would also like to acknowledge the amazing friends who have always been there for me – Alessandro, Gaia, Salvatore, and the entire A.A. group, for sharing countless adventures and experiences with me – as well as my university buddies, Federico and Dino – for supporting each other, making this academic journey much smoother. To Monica, you have been my lifeline when challenges felt insurmountable, and I will forever be grateful for your unwavering presence and support.

The support and love from each of you have had a profound impact on my academic and personal growth, and I can't thank you enough.

*This thesis helped to plant a tree in Kenya through the Treedom project.*

treedom.net/it/user/marco-calenda-3171/trees/M77-EY75