
**Università degli Studi di Salerno**

Dipartimento di Informatica

Dottorato di Ricerca in Informatica
Curriculum Computer Science and Information Technology
XXXVII Ciclo

Tesi di Dottorato / Ph.D. Thesis

# Detecting Non-Functional Requirements by exploiting NLP-based technologies and techniques

**Francesco CASILLO**

| | |
|---|---|
| Supervisor: | **Prof. Carmine GRAVINO** |
| Co-Supervisor: | **Prof. Vincenzo DEUFEMIA** |
| PhD Program Director: | **Prof. Andrea DE LUCIA** |

A.A. 2023/2024

## ABSTRACT

The management of Non-Functional Requirements (NFRs) plays an important role on the quality and stability of software systems. NFRs, which cover aspects such as privacy, security, fairness, and performance, are often challenging to identify and address during early phases of software development. Failure to properly detect and manage these requirements can lead to significant security vulnerabilities, privacy violations, and performance bottlenecks, which may not be immediately evident but can severely affect the system post-deployment. Despite their importance, NFRs are frequently implicit in requirement documents, making their extraction from unstructured text a complex task. In this context, the automation of NFR detection is an essential step forward to assist developers in recognizing and addressing these requirements efficiently, especially in agile environments where time and expertise may be limited.

The goal of this thesis is to propose and assess a range of solutions to automate the detection and classification of NFRs from software requirements documents. By leveraging state-of-the-art Natural Language Processing (NLP) techniques, Machine Learning (ML) and Deep Learning (DL) approaches, this work aims to tackle the problem of NFR identification from multiple perspectives, providing tailored solutions for the different types. The objective is to create a set of methodologies that can detect various NFRs in their early stages, mitigating risks related to privacy, security, fairness, and other non-functional aspects of software systems.

The first NFR analyzed is privacy, which has become a pressing concern in software engineering due to increasing data protection regulations, such as the General Data Protection Regulation (GDPR). This thesis introduces a Deep Learning-based approach that combines traditional NLP techniques with Transfer Learning to automatically identify privacy requirements when the latter are defined as User Stories. Specifically, Convolutional Neural Networks (CNNs) are employed to extract semantic features from textual descriptions,

allowing the model to recognize privacy-related information even when it is not explicitly stated. The system was tested on a dataset of User Stories and demonstrated its ability to detect privacy issues with a high degree of Accuracy, providing developers with crucial insights early in the development lifecycle.

The second aspect focuses on security requirements, a vital NFR given the growing threats of cyber-attacks and system vulnerabilities. Addressing the challenge of identifying security-related requirements in unstructured documents, in this work domain-specific approaches are contrasted with advanced domain-independent solutions. The research shows that while shallow ML models can be effective in specific contexts, Transformer-based models significantly improve the generalizability of security requirement detection across different software projects. This approach ensures that security considerations are not overlooked, even when requirements are vague or incomplete and independently from the software application context.

With the increasing integration of ML algorithms in decision-making systems, fairness has emerged as a novel NFR. ML systems can reproduce biases present in historical data, leading to unethical outcomes. This thesis explores fairness requirements by introducing ReFAIR, a context-aware recommender system designed to detect fairness concerns from User Stories. Using NLP and Word Embedding techniques, ReFAIR analyzes the domain and task of the system being developed to identify sensitive attributes and potential sources of bias. This approach provides developers with early-stage recommendations on how to address fairness issues, contributing to the development of fair and unbiased ML systems.

Lastly, the thesis investigates the use of Quantum Natural Language Processing (QNLP) for the classification of multiple NFRs, including performance, usability, and operability. As an exploratory study, this research examines the potential of quantum computing to handle the linguistic complexity of NFRs with smaller datasets, a limitation of many traditional NLP approaches. By comparing shallow ML models using Word Embeddings (e.g., GloVe, Word2Vec) with quantum circuit-based models, the study highlights the advantages and current limitations of quantum methods in NFRs detection. While ML models still outperform quantum approaches in terms

of performance, QNLP offers promising insights, particularly in handling NFRs classification tasks where data volume is an issue.

Through the exploration of these diverse approaches, this thesis contributes to the field of Requirements Engineering by providing automated NLP-based solutions for the detection of key NFRs. These findings pave the way for future work in automating NFR identification using diverse technologies, including quantum computing, to ensure that non-functional aspects are considered into software systems from the earliest stages of development.

# CONTENTS

# LIST OF FIGURES

## LIST OF TABLES

# PART I

# INFERNO

*"Nel mezzo del cammin di nostra vita*
*mi ritrovai per una selva oscura,*
*ché la diritta via era smarrita.*

*Ahi quanto a dir qual era è cosa dura*
*esta selva selvaggia e aspra e forte*
*che nel pensier rinova la paura!*

*Tant'è amara che poco è più morte;*
*ma per trattar del ben ch'i' vi trovai,*
*dirò de l'altre cose ch'i' v'ho scorte.*

*Io non so ben ridir com'i' v'intrai,*
*tant'era pien di sonno a quel punto*
*che la verace via abbandonai."*

*Dante Alighieri, Inferno, Canto I, vv. 1 - 12.*

# 1 THE NEED FOR AUTOMATING NON-FUNCTIONAL REQUIREMENTS DETECTION

Requirements Engineering (RE) phase in software development revolves around the elicitation, analysis, and specification of Functional and Non-Functional requirements (NFRs) that a software system must guarantee to its users [24, 164, 202]. While Functional Requirements (FR) describe the system's functionalities, NFRs define how well the system should perform, focusing on quality attributes like security, privacy, performance, usability, and so on [44, 167]. Despite their critical importance, NFRs are often under-prioritized during early development stages due to their abstract nature [160, 229], even though they impact the quality of software systems, ensuring they meet security, privacy, and performance standards [204]. Lacks in considering NFRs can result in severe consequences, such as security vulnerabilities or privacy breaches, which can lead to financial losses, system failures, and legal implications [13, 135]. For example, the infamous Equifax data breach[1] highlighted the catastrophic results of overlooking security NFRs. Similarly, privacy requirements have gained prominence with the enforcement of the General Data Protection Regulation (GDPR), necessitating strong privacy considerations during system design [191]. Furthermore, the rise of AI-driven systems has amplified the need for fairness as an NFR, particularly in avoiding biases that could perpetuate social inequalities [206].

Ensuring the early detection and management of NFRs is therefore vital for reducing risks and ensuring long-term system sustainability. Delaying the identification of NFRs until later stages, such as testing or deployment, can lead to substantial costs due to reengineering and refactoring [229]. For example, it is estimated that fixing a security vulnerability after release can be up to 30 times more expensive than addressing it during the design phase [18].

1 The Equifax Data Breach: An Overview and Issues for Congress. Accessed October 2024. https://crsreports.congress.gov/product/pdf/IN/IN10792

Thus, the earlier NFRs are detected, the fewer resources are required to address them, as issues can be resolved before they affect critical aspects of the system. Despite this, traditional methods of NFR elicitation remain heavily dependent on manual processes such as stakeholder interviews and document analysis, which are time-consuming, error-prone, and unsuitable for large-scale systems [40].

Given the growing scale and complexity of modern software systems, relying solely on manual approaches for NFR detection is no longer sustainable and efficient. This has led interest in automated approaches based on Natural Language Processing (NLP), which can efficiently process and extract NFRs from large volumes of text, such as User Stories (USs) and system specifications [195].

## 1.1 RESEARCH GAPS

While Machine Learning (ML) and NLP-based approaches have shown promise in automating NFR detection, they come with several limitations. Traditional ML models like Support Vector Machines (SVMs) have been used for classifying NFRs but struggle with the implicit and vague language used in NFR documentation [16, 124]. These methods also require substantial amounts of labeled data, which can be difficult to obtain in real-world development environments [180].

Many existing tools are predominantly focused on functional requirement verification, leaving non-functional aspects like privacy, security, and performance either undetected or relegated to manual review [75]. Furthermore, existing NLP models struggle to capture the implicit nature of NFRs, often leading to incomplete or inaccurate detection results [100, 136]. Indeed, existing approaches rely on basic keyword matching techniques or statistical lexical evidence, which fail to capture the complex semantics and trade-offs between different NFRs. For example, balancing security with usability often requires nuanced understanding that cannot be captured by simple keyword searches [103]. This gap is particularly pronounced in domains where NFRs are critical, such as healthcare, financial systems,

and AI-driven applications, where missing a key NFR can result in catastrophic failures or unexpected system behaviors [111, 158, 165].

Deep Learning (DL) techniques, including Transformer-based architectures like BERT (Bidirectional Encoder Representations from Transformers) [56] and GPT (Generative Pre-trained Transformers) [176], have demonstrated substantial improvements in NLP tasks. However, when applied to the detection of NFRs, these models face certain limitations. While Transformers excel in capturing contextual embeddings and improving language understanding, they still struggle with the implicit and abstract nature of NFRs, which are often hidden in vague or incomplete requirements documents [33]. For example, while a functional requirement might be explicitly stated as "The system shall allow users to log in using a username and password", the associated security NFR might only be implied through references to "Sensitive data" or "Access control". Transformers struggle to detect such implicit NFRs without extensive fine-tuning or domain-specific pre-training.

Another challenge is that DL models may require large, annotated datasets to perform effectively. In the case of NFRs, datasets are scarce, and even when they are available, the requirements are context-dependent and domain-specific, making it difficult for transformers to generalize across different projects or industries [180].

Additionally, Transformer-based models are computationally expensive and may overfit on smaller datasets. In the context of NFR detection, where annotated examples are limited, this may lead to poor generalization, particularly when dealing with the diverse linguistic structures present in real-world requirements documents [218]. For example, a system might need to be "secure" or "performant", but the specific mechanisms or metrics that define security or performance are often missing. Current approaches that rely on keyword-based approaches are unable to handle this semantic complexity, resulting in either an over-reliance on shallow pattern recognition or missed opportunities to detect critical NFRs [229]. Moreover, the documentation available for NFRs is sparse or incomplete, making it difficult for traditional models to perform well.

Lastly, the generalizability of existing NFR detection models is another key gap. Many tools and approaches are tailored to specific

domains, such as enterprise applications or consumer-facing web services, but struggle to generalize to other sectors like embedded systems, critical infrastructure, or high-assurance AI applications [180]. This lack of generalizability limits the effectiveness of automated NFR detection across the wide variety of software systems being developed today. New methods are needed to build domain-agnostic models that can accurately detect NFRs across various industries and application types without significant reconfiguration or retraining.

In summary, the main gaps found in literature are the following:
⚠ **Gap 1**: The language structure by which the requirements are defined is not much considered.
⚠ **Gap 2**: Basic keyword matching or statistical lexical frequencies techniques fail to capture the complex semantics and trade-offs between different NFRs.
⚠ **Gap 3**: The best performances have been achieved with high computational cost solutions.
⚠ **Gap 4**: Limited amount of labeled data.
⚠ **Gap 5**: The approaches lack of generalizability across domains.

## 1.2 RESEARCH STATEMENT

The GOAL of this thesis is to design, implement and evaluate new solutions and technologies to address the task of NFRs detection with the aim of broadening the spectrum of viable solutions in modern development contexts. The PURPOSE is to provide engineers with new ways to deal with NFRs as early as possible in software development to reduce the cost and effort required if they are not properly considered in the requirements analysis phase. The research is conducted taking into account the PERSPECTIVE of both practitioners and researchers. In particular, practitioners are interested in the suitability, usability and integrability of such automated approaches in their workflow, already based on different technologies and methodologies. Researchers, on the other hand, are interested in expanding the information that can be exploited to improve efficiency and accuracy in detecting NFRs, thus expanding the pool from which to extract knowledge according to the quality of the software being detected.

This thesis achieves its goal by answering three high-level research questions (RQs), each one representing a pillar of this research and filling the research gaps mentioned above:

🔍 **RQ**$_A$ To what extent can analyzing the structure of requirement language enhance the detection of NFRs?

🔍 **RQ**$_B$ How much can the availability of hardware resources counterbalance the data scarcity?

🔍 **RQ**$_C$ To what extent can we introduce domain agnostic NLP-based technologies and techniques?

While the RQs are broadly applicable to various NFRs, this dissertation focuses on privacy, security, and fairness. These NFRs were chosen based on their increasing relevance in recent years, as identified in the Systematic Literature Review conducted in this study [32]. Privacy and security have long been critical concerns in software development, reinforced by regulations such as GDPR, while fairness is an emerging yet under-explored NFR in the context of RE. The lack of established methods for supporting fairness requirements, coupled with its growing significance in AI-based decision-making systems, justifies its inclusion alongside privacy and security.

The selection of multiple NFRs, rather than a singular focus, is motivated by the need to evaluate NLP-based solutions across different types of NFRs. This allows for a broader assessment of how such techniques perform in varying contexts—ranging from explicit, well-documented NFRs like security to more implicit and context-sensitive ones like fairness. Furthermore, by considering multiple NFRs, this dissertation explores the NFRs detection as a multi-class classification task, where different NFRs may coexist or interact within the same requirements specification.

Part II of this dissertation presents different methodologies applied to distinct NFRs. Each chapter corresponds to a specific scenario and is mapped to its respective publication, a mapping that is explicitly detailed in the following section.

## 1.3 THESIS STRUCTURE AND CONTRIBUTIONS

This dissertation advances the literature in NFRs detection in different manners. Firstly, in Part I, namely Inferno, it introduces the problem with its importance, limitations and how this research face it in Chapter 1, to then summarizing the state-of-the-art in Chapter 2, providing an overview of the approaches proposed when dealing with the NFRs classification task, privacy and security requirements identification and fairness consideration in early software development stages.


Figure 1.1: Mapping each chapter to each document and RQs addressed.

Then, in Part II, namely Purgatorio, it reports novel solutions when dealing with privacy concerns, fairness considerations and security issues, respectively presented in Chapters 3, 4, 5, each one reported in a scientific publication **[J1, C3, J3]**. In Chapter 6 is presented an attempt to NFRs detection when modeled as a multiclass classification task, introducing a novel method to represent requirements **[J4]**.

Lastly, in Part III, namely Paradiso, it presents a discussion on the lesson learned, how to take advantage of such findings and open issues that need to be addressed in Chapter 7. Chapter 8 concludes the thesis. All the research presented in this thesis supports open science by making all experimental materials publicly accessible. This includes raw data, reusable datasets, and reproducible scripts, which are provided through online appendices. Additionally, pre-prints of the published articles are made available online.

**Main Contributions** In the following are summarized the research works discussed in this thesis and how they contributed to address the problems described in previous sections. The thesis makes four key contributions, each mapped in the relative Chapter (Ch.):

**Ch. 3 Detecting privacy requirements from User Stories with NLP transfer learning models**

An empirical study to assess the effectiveness of an automated approach to identify privacy requirements embedded in agile USs. By utilizing Transfer Learning, this research enhances detection accuracy by transferring knowledge from a privacy-sensitive detector in unstructured text sources. In particular, the approach combines NLP techniques aimed to inform a CNN with entity tokens, syntactic structure and parts of speech, and with a privacy lexicon to capture implicit privacy concerns in different contexts, providing an effective solution for privacy-aware requirements engineering in agile development environments. Empirical evaluation demonstrates a significant performance improvement over traditional methods when considering the language structure features. This work answers $\mathbf{RQ}_A$ and $\mathbf{RQ}_C$, addressing **Gap 1**, **Gap 2** and **Gap 5**. This study has been published in Information and Software Technology (IST) journal.

**Ch. 4 ReFAIR: Toward a Context-Aware Recommender for Fairness Requirements Engineering**

An implementation of a context-aware framework designed to support fairness requirements engineering. ReFAIR uses NLP and word embeddings to classify sensitive features from USs by identifying the application domains of the system and the ML tasks or AI-based solutions intended to be integrated or implemented. This innovative framework helps requirements engineers address fairness concerns early in the development lifecycle by recommending sensitive features that are contextually relevant to each project. Empirical validation shows ReFAIR's high accuracy, demonstrating its potential as a practical tool for considering fairness already into RE phases. This work answers $\mathbf{RQ}_B$ and $\mathbf{RQ}_C$, addressing **Gap 2**, **Gap 3** and **Gap 4**. This study has been presented at 46th International Conference on Software Engineering (ICSE '24).

**Ch. 5 Beyond Domain Dependency in Security Requirements Identification**

A large empirical study on the effectiveness of different methodologies based on both shallow ML and advanced BERT-based models to automatically detect security requirements across different domains. By leveraging word embeddings and ML algorithms or pre-trained models on datasets like Common Weakness Enumeration (CWE) and Common Vulnerabilities and Exposures (CVE), this work tries to analyze and achieve domain independence, allowing it to accurately classify security requirements without relying on extensive domain-specific data. Empirical results demonstrate the model's superior performance in both intra-domain and inter-domain contexts, providing a significant step forward in automated, adaptable security requirements detection. This work answers $\mathbf{RQ}_B$ and $\mathbf{RQ}_C$, addressing **Gap 3**, **Gap 4** and **Gap 5**. This study has been published in Information and Software Technology (IST) journal.

**Ch. 6 A First Eye on the Impact of Quantum Natural Language Representations for Non-Functional Requirements Classification**

An experimentation of novel textual representations techniques with the aim to explore the potential of Quantum Natural Language Processing for classifying NFRs. This work introduces three novel approaches: shallow ML models with word embeddings, tensor network-based representations using string diagrams, and quantum circuit-based models. Each method is tested on a multiclass NFR classification task, showing that while traditional approaches based on word embeddings and ML algorithms still lead in performance, string diagrams and quantum models demonstrate promising potential for handling complex linguistic structures, opening new pathways for integrating such solutions in RE processes. This work answers $\mathbf{RQ}_A$ and $\mathbf{RQ}_B$, addressing **Gap 1**, **Gap 3** and **Gap 4**. This study is currently under major revision in Transactions on Software Engineering and Methodology (TOSEM).

# LIST OF PUBLICATIONS

The complete list of publications is reported below. The articles whose title is in **boldface** were discussed in this dissertation.

## INTERNATIONAL JOURNAL PAPERS


[J1] Francesco Casillo, Vincenzo Deufemia, and Carmine Gravino. "**Detecting privacy requirements from User Stories with NLP transfer learning models**." In: *Information and Software Technology* 146 (2022), p. 106853. ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2022.106853.

[J2] Francesco Casillo, Vincenzo Deufemia, and Carmine Gravino. "**Automatic Identification of Privacy and Security Requirements: A Systematic Literature Review**." In: *Under minor revision in Requirement Engineering Journal* (2025).

[J3] Francesco Casillo, Vincenzo Deufemia, and Carmine Gravino. "**Beyond Domain Dependency in Security Requirements Identification**." In: *Information and Software Technology*, 182 (2025), p. 107702. ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2025.107702.

[J4] Francesco Casillo, Vincenzo Deufemia, Fabio Palomba, and Carmine Gravino. "**A First Eye on the Impact of Quantum Natural Language Representations for Non-Functional Requirements Classification**." In: *Under major revision in Transactions on Software Engineering and Methodology (TOSEM)* (2025).

[J5] Gianmario Voria, Francesco Casillo, Carmine Gravino, Gemma Catolino, and Fabio Palomba. "RECOVER: Toward the Automatic Requirements Generation from Stakeholders' Conversations." In: *Under major revision in Transactions on Software Engineering* (2025).

### INTERNATIONAL CONFERENCE PAPERS

[C1] Francesco Casillo, Vincenzo Deufemia, and Carmine Gravino. "PReDUS: A Privacy Requirements Detector From User Stories." In: *REFSQ Workshops*. 2022.

[C2] Francesco Casillo, Antonio Mastropaolo, Gabriele Bavota, Vincenzo Deufemia, and Carmine Gravino. "Towards Generating the Rationale for Code Changes." In: *33rd IEEE/ACM International Conference on Program Comprehension (ICPC 2025)*. ICSE '25. Ottawa, Canada, 2025.

[C3] Carmine Ferrara, Francesco Casillo, Carmine Gravino, Andrea De Lucia, and Fabio Palomba. "**ReFAIR: Toward a Context-Aware Recommender for Fairness Requirements Engineering**." In: *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. ICSE '24. Lisbon, Portugal: Association for Computing Machinery, 2024. ISBN: 9798400702174. DOI: 10.1145/3597503.3639185.

# 2 BACKGROUND

Understanding and addressing NFRs is essential for developing robust, secure, and ethically responsible software systems. Unlike functional requirements that specify the operations a system must perform, NFRs define quality attributes that a system must satisfy, such as performance, security, privacy, and fairness. As software systems become more complex and pervasive, the need for effective NFR management has become more pressing. This chapter provides an exploration of the classification of NFRs and a focused discussion on three types: privacy, security, and fairness. Additionally, it reviews existing methodologies and frameworks that have been developed to automate the identification of these NFRs, paving the way for a better understanding of the research contributions of this thesis.

## 2.1 NON-FUNCTIONAL REQUIREMENTS CLASSIFICATION

In recent years, numerous researchers have contributed to the classification of non-functional requirements (NFRs). Some studies have focused on identifying a comprehensive set of NFR categories, while others have developed automated methods to classify these requirements. This section provides an overview of the most significant contributions in this field.

K. Rahman et al. [178] investigates the classification of NFRs into categories such as security, usability, operability, maintainability, and performance. The study introduces a hybrid deep learning framework that combines LSTM (Long Short-Term Memory) and BiLSTM (Bidirectional LSTM) models with Artificial Neural Networks (ANNs). It uses Word2Vec for feature extraction and vectorization, enhancing the ability of the models to learn contextual representations of words in the NFR corpus. The research utilized a combined dataset of 914 NFRs from two publicly available sources (PROMISE_exp [59] and

PROMISE-NFR [46]) and conducted extensive preprocessing. The experimental results demonstrate that the hybrid models (LSTM-ANN and BiLSTM-ANN) significantly outperform their counterparts. The LSTM-ANN model achieved an F1-score ranging from 58% to 82%, while the BiLSTM-ANN model attained F1-scores up to 88%. Overall, the BiLSTM-ANN model showed performance with improvements in precision, recall, and F1-scores compared to simple LSTM and BiLSTM models, with micro-average F1-scores reaching 78%.

F. Khayashi et al. [101] focuses on distinguishing between Functional Requirements (FRs) and NFRs. The NFRs analyzed include general quality factors such as usability, reliability, security, and performance. The study employs a variety of DL models: LSTM, BiLSTM, GRU, BiGRU, and CNN. It also incorporates ensemble methods through hard and soft voting mechanisms. For feature extraction, the paper uses two word embedding techniques: one based on the Keras library trained on the dataset itself and another using GloVe embeddings trained on a large-scale external corpus. The preprocessing steps involve text cleaning using the NLTK library, including removing punctuation, converting text to lowercase, and eliminating stop words. The dataset utilized is a processed version of the PURE repository [88], comprising 4,661 requirements, with 2,617 labeled as functional and 2,044 as non-functional. The models were evaluated using metrics like precision, recall, and F1-score. Results showed that the BiLSTM model had the highest performance among individual models, achieving an F1-score of 79% with Keras embeddings. The ensemble methods improved results further, with hard voting yielding the best overall performance, reaching an F1-score of 80%.

G. Li et al. [116] investigate the classification of both FRs and NFRs, focusing specifically on NFR categories like usability, security, operability, and performance. The proposed method, DBGAT (Dependency-BERT Graph Attention Network), integrates BERT for initial word embeddings to capture rich contextual information and a Graph Attention Network (GAT) to incorporate syntactic and structural features of requirements sentences. The approach constructs dependency parse trees, transforming them into graphs where nodes represent words, and edges capture syntactic relationships. This structure allows the model to learn deep semantic representations,

enhancing classification performance. The study uses the PROMISE dataset [46], consisting of 625 requirement statements, split into 255 FRs and 370 NFRs, and the Concordia RE corpus, which contains 3,064 annotated sentences. The datasets cover various software requirements, testing the model's generalization and performance. The DBGAT model achieves outstanding results, with an F1-score of up to 91% for seen projects and 88% for unseen projects.

B. Li et al. [114] categorize NFRs such as performance, security, usability, maintainability, and reliability, introducing a DL model named NFRNet, which incorporates an improved BERT model using N-gram masking for word embeddings and a Bi-LSTM network to capture contextual information. The approach involves extensive preprocessing of textual requirement descriptions, including normalization, stop word removal, and lemmatization. The improved BERT model enhances the representation of context by considering both character-level and phrase-level dependencies using N-gram masking. The Bi-LSTM network then synthesizes these embeddings to classify NFRs. The model also employs multi-sample dropout for regularization, which improves generalization and reduces training iterations. The research uses an expanded dataset called SOFTWARE NFR, derived from the original PROMISE dataset [46], increasing from 370 to 6,222 requirement descriptions across 32 NFR categories. The proposed NFRNet model achieves superior performance compared to 17 benchmark models, with a Precision of 91%, a Recall of 92%, and an F1-score of 91%, demonstrating significant improvements in the automated classification of NFRs.

K.M. Ashikur Rahman et al [179] propose a comprehensive framework that includes two models: DReqANN (Deep Requirement Artificial Neural Network) and DReqBiLSTM (Deep Requirement Bidirectional Long Short-Term Memory). The methodology involves preprocessing using Natural Language Toolkit (NLTK) for tasks like stop word removal and lemmatization, followed by TF-IDF for feature extraction and word embeddings for semantic understanding. The models use multi-layer architectures with dropout regularization to prevent overfitting and optimize performance. The study utilizes a dataset of 914 NFR instances derived from the PROMISE_NFR [46] and PROMISE_exp [59] datasets, categorized into five primary NFR

types. The DReqANN model achieved precision between 81% and 99.8%, recall between 74% and 89%, and an F1-score ranging from 83% to 89%, outperforming traditional approaches and demonstrating the efficacy of DL techniques for automated NFR classification.

M. Akter Metu et al. [144] address the classification of NFRs including availability, security, usability, maintainability, and performance. The research proposes a hybrid deep learning model combining SVM with Bi-LSTM. The Bi-LSTM component captures contextual and sequential dependencies in requirement sentences, while SVM is employed for the output layer to handle high-dimensional data and improve classification performance. The text preprocessing involves tokenization, data cleaning, and normalization, and feature extraction is achieved using word embeddings integrated into a Keras embedding layer. The model was trained and evaluated on the PROMISE_exp dataset [59], which includes 969 software requirement sentences distributed over 12 requirement classes, covering functional and non-functional requirements. The hybrid model achieved an accuracy of 98%, outperforming several baseline models, including traditional DL approaches. The results highlight the model's robustness and effectiveness, validated using 10-fold cross-validation.

## 2.2 PRIVACY AND SECURITY REQUIREMENTS IDENTIFICATION

Interest in the identification of specific NFRs such as privacy and security has become increasing in recent years. While researchers and practitioners have provided solutions for the identification of such NFRs in approaches that also include other types of NFRs, as we have seen in the previous section, there have also been attempts to identify NFRs considered individually, then methodologies designed to specifically identify security, privacy, and so on. One of the contributions of this thesis is a systematic review of the literature (SLR) on automatic identification of security and privacy requirements **[J2]**. Below, the main findings from the study are summarized.

The field of security and privacy requirement identification has seen extensive research, focusing primarily on ML-based techniques to automate the identification of these requirements.

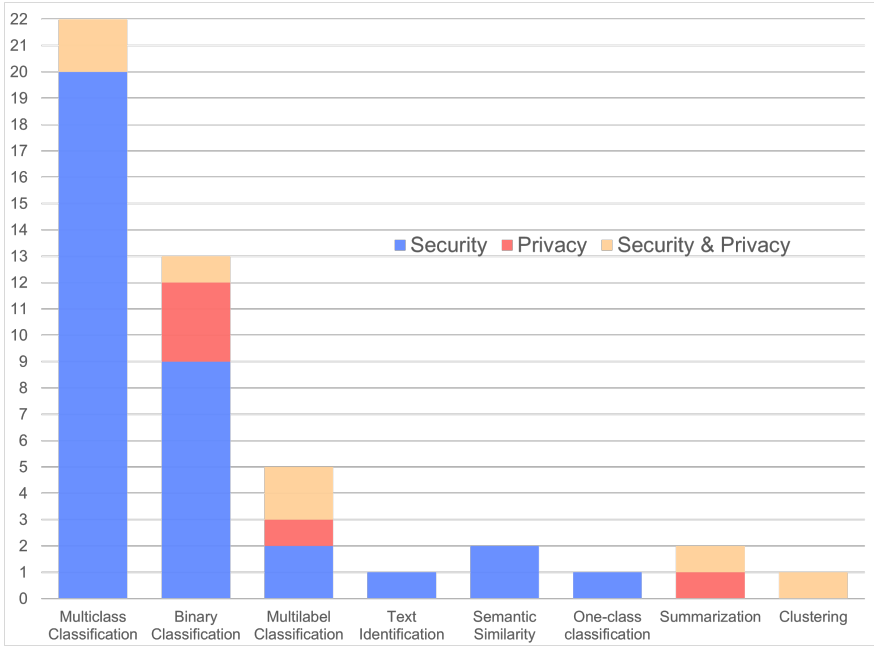


Figure 2.1: Number of papers for the type of task and problem addressed.

Figure 2.1 illustrates that the most common approaches include multi-class and binary classification, with multi-class classification being the predominant approach in the literature. This method, used by 20 studies, mainly targets security requirements across various non-functional requirements (NFRs) [3, 4, 10, 39, 47, 79, 84, 91, 98, 102, 109, 115, 116, 123, 131, 181, 183, 200, 214, 217], while binary classification, though less common, is primarily applied for specific privacy or security requirements [71, 103, 106, 115, 118, 119, 150, 157, 194, 199, 225, 242].

In terms of techniques, SVM, TF-IDF, and Naive Bayes (NB) are the most frequently utilized for classification tasks, appearing in over a dozen studies [4, 62, 91, 102, 115, 123, 138, 150, 153, 199, 200, 210, 214, 237, 242]. Recently, DL models such as BERT and Word2Vec have gained prominence, reflecting an increased focus on leveraging advanced NLP methods for enhanced accuracy and attempts for context-aware requirement extraction [39, 79, 114, 116, 131, 190, 217].

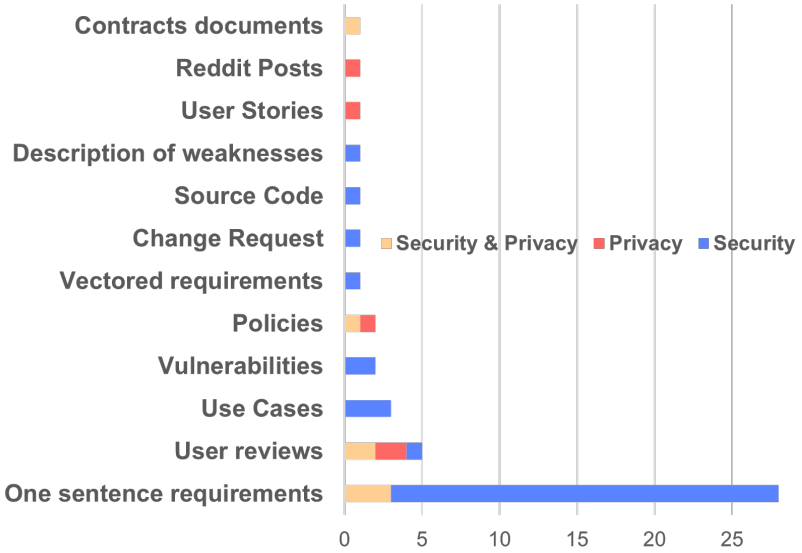


Figure 2.2: Employed document types.

The SLR investigate then the types of documents utilized, with Figure 2.2 illustrating their distribution across the literature.

The analysis reveals that concise, single-sentence requirements are the most frequently used document type, appearing in 28 studies, with a predominant focus on security requirements [3, 4, 10, 39, 47, 79, 91, 98, 102, 103, 106, 109, 114, 116, 118, 131, 134, 150, 181, 183, 188, 194, 200, 214, 217, 225, 236, 237]. This reflects a preference for straightforward requirement statements in the automated identification process.

Other document types include USs, Reddit posts, contract documents and policies, which are mainly used for privacy requirements, underscoring the role of formal guidelines or user perspectives in addressing these concerns [29, 119, 137, 190]. Additionally, Use Cases and vulnerabilities are frequently associated with security requirements, highlighting their relevance in capturing potential risks during the requirements analysis phase [71, 83]. This variety demonstrates the need to leverage multiple sources to build comprehensive models for security and privacy requirement identification.

An interesting aspect that emerged from the SLR is that there is a lack of widely used datasets when it comes to identifying privacy or privacy and security together. In fact, each work uses a different dataset as well as a different type of requirement [29, 62, 114, 119, 134, 137, 138, 157, 188, 190, 210, 242]. The situation changes when it comes to identifying security requirements, where the PROMISE dataset [46] is used in several works [3, 4, 10, 39, 47, 79, 91, 98, 109, 116, 118, 131, 181, 183, 200, 214, 236, 237], becoming a baseline to be used to compare with other approaches. Next, there is the SecReq dataset, which includes requirements from 3 different projects (Common Electronic Purse (ePurse), Customer Premises Network (CPN) and the Global Platform Specification (GPS)) [103, 106, 118, 153, 194, 217] and a dataset of about 40 use cases with non-functional requirements related to an application called iTrust [71, 199, 200, 234].

In the end, investigating effective methods for identifying privacy and security requirements, the chosen approaches vary significantly based on the document types analyzed, such as one-sentence requirements, use cases, user reviews, and vulnerabilities.

For one-sentence requirements, the NFRNet model by Li et al. [114] demonstrates outstanding performance in multiclass classification. NFRNet combines a BERT word embedding layer with a BiLSTM network to classify requirements into 32 NFRs categories, achieving an impressive F1-score of 91.5%. Similarly, for security requirements, Luo et al. [131] introduce PRCBERT, a prompt learning approach based on BERT that yields a remarkable F1-score of 98% exploiting PROMISE [46]. These results highlight the effectiveness of neural network-based approaches in one-sentence requirement analysis.

When analyzing Use Cases, the best approaches focus on security requirements. Gärtner et al. [71] propose a knowledge-based approach leveraging security knowledge extraction, achieving 98% accuracy on the iTrust dataset. Alternatively, Slankas et al. [199] apply a combination of supervised learning models using Stanford Type Dependency Representation and a voting system with K-Nearest Neighbors (KNN), NB, and SVM, yielding an F1-score of 89%. Despite the effectiveness of both methods, their differing metrics complicate direct comparison, indicating that knowledge-based and ML approaches can both be effective depending on the evaluation criteria.

For user reviews, effective methods vary by task. In identifying both security and privacy requirements, Nguyen et al. [157] employ SMOTE, Bag-of-Words, and SVM, achieving an AUC of 93%. For security-focused review summarization, Tao et al. [210] achieve an F1-score of 89% using TF-IDF and SRR-Miner. Privacy concerns from user reviews are effectively extracted by Ebrahimi et al. [62], using a combination of TF-IDF, GloVe embeddings, and LDA with a precision of 75%. Different metrics again preclude a single "best" approach, but hybrid and embedding-based techniques demonstrate strong results for both security and privacy concerns in user reviews.

The task of identifying security requirements in the context of vulnerabilities is challenging. Imtiaz et al. [83] use Soft Ensemble methods achieving modest precision and recall scores (13.6% and 64.1%, respectively). Another study by Imtiaz et al. [84] applies VSM and LSI, yielding varying precision and recall, with lower scores overall. The relatively low performance suggests that vulnerability detection from documents may require more robust or novel approaches to improve accuracy.

In summary, document type greatly influences method efficacy. Multiclass classification methods, such as NFRNet and PRCBERT, excel in one-sentence requirements. For complex documents like user reviews and Use Cases, tailored approaches using hybrid and ensemble techniques perform well. However, vulnerability identification remains challenging, underscoring the need for specialized methodologies in that domain.

## 2.3 FAIRNESS AS NON-FUNCTIONAL REQUIREMENT

Fairness as a NFR is a rapidly evolving area of study, gaining interest as ML systems are increasingly deployed in sensitive, real-world contexts. Fairness in these systems typically relates to preventing discrimination against specific groups or individuals.

Fairness definitions are diverse, each tailored to specific ethical or regulatory needs. Verma and Rubin [222] categorize fairness definitions into three groups: (1) statistical fairness, ensuring consistent predictions across sensitive groups, such as "Statistical Parity"; (2)

individual fairness, maintaining fairness among individuals through approaches like "Fairness through Unawareness"; and (3) causal fairness, focusing on causal relationships, such as "Counterfactual Fairness".

Similarly, Mehrabi et al. [141] define fairness in terms of group-based and individual-based reasoning, introducing concepts like "Equalized Odds" for group fairness and "Fairness through Awareness" for individual considerations.

Bias mitigation strategies can be based on data preprocessing, aiming to modify training data to align with fairness principles. Biswas and Rajan [17] evaluated 37 different preprocessing pipelines across datasets, demonstrating that data transformations can positively impact fairness. Nargesian et al. [155] proposed data augmentation techniques to enhance fairness, while Moumoulidou et al. [152] explored data diversity under fairness constraints, showing that balanced representation across groups can mitigate model bias. Another notable approach, namely "Fair-SMOTE", developed by Chakraborty et al. [36], balances training data by oversampling underrepresented sensitive groups, improving fairness in model outcomes.

Beyond data preprocessing, some studies address fairness through ML model optimization, often seeking trade-offs between fairness and accuracy. Hort et al. [81] introduced "Fairea", a mutation-based approach that benchmarks bias mitigation techniques based on fairness-accuracy trade-offs. Similarly, Chen et al. [42] proposed "MAAT", an ensemble learning approach that improves fairness without sacrificing performance. Galhotra et al. [70] developed "Themis", a test generation tool that highlights and addresses data-driven unfairness through resampling strategies aligned with fairness metrics.

Despite advancements in fairness-aware data preprocessing and model tuning, fairness as an NFR remains under-explored in requirements engineering. Recognizing this gap, Soremekun et al. [205] emphasize the importance of considering fairness early in the requirements specification phase, highlighting the lack of standardized frameworks for integrating ethics into requirements engineering [15]. This lack of standards makes it challenging to address fairness definitions within a single requirements specification, as each fair-

ness notion might require distinct considerations depending on the application domain and ML task [26].

Context sensitivity further complicates fairness requirements in ML, as ethical concerns can vary greatly depending on the domain and the specific functionalities required. Previous work [35, 69] underscores the difficulty of optimizing fairness across different application domains, as each context may necessitate a unique fairness approach. For instance, fairness requirements in healthcare might vastly differ from those in finance or criminal justice, making it difficult to adopt a one-size-fits-all strategy.

In a broader perspective, AI researches provide preliminary tools, such as ethical content analyzers for social media posts [82], offering analytics on ethical considerations in textual data [27, 37, 80, 125].

Although these methods provide valuable solutions, they overlook the potential impact of early-stage requirements engineering. Soremekun et al. [205] argue that making practitioners aware of sensitive features during requirements elicitation could significantly influence the entire ML lifecycle. Early recommendations on sensitive features could guide subsequent stages of development, from data collection to model training, enabling stakeholders to adopt more fairness-aware practices throughout the ML pipeline.

# PART II

# PURGATORIO

*"Per correr miglior acque alza le vele*
*omai la navicella del mio ingegno,*
*che lascia dietro a sé mar sì crudele;*

*e canterò di quel secondo regno*
*dove l'umano spirito si purga*
*e di salire al ciel diventa degno.*

*Ma qui la morta poesì resurga,*
*o sante Muse, poi che vostro sono;*
*e qui Calïopè alquanto surga,*

*seguitando il mio canto con quel suono*
*di cui le Piche misere sentiro*
*lo colpo tal, che disperar perdono."*

*Dante Alighieri, Purgatorio, Canto I, vv. 1 - 12.*

# 3

# DETECTING PRIVACY REQUIREMENTS FROM USER STORIES WITH NLP TRANSFER LEARNING MODELS

## 3.1 INTRODUCTION

RE, focusing on the documentation and management of system requirements, remains a complex activity where misunderstandings can lead to costly design flaws and system failures [172, 203]. This complexity is accentuated in Agile methodologies, where requirements are expressed in natural language through User Stories (USs) that improve collaboration and flexibility. Despite the advantages of Agile approaches, identifying NFRs, such as privacy requirements, remains a challenging task, often due to the limited expertise of stakeholders and the informal nature of the requirements [66].

USs follow a structured format characterized by the *who*, the *what*, and the *why* of a requirement, becoming a de facto standard for requirement specification [129]. For example, a typical US might read: *"As a site member, I want to access the Facebook profiles of other members so that I can share my experiences with them."* [50]. This format introduces challenges in ensuring that important NFRs such as privacy and security are explicitly addressed and well-defined. Several frameworks and methodologies have been proposed for analyzing the syntactic structure of USs to improve their clarity and accuracy [77, 90, 127], while others focus on transforming USs into models and artifacts for subsequent software development stages [63, 95, 130]. Yet, little attention has been paid to systematically addressing NFRs such as privacy within USs during the early phases of Agile development.

In the context of privacy requirements, many efforts have been devoted to privacy disclosure, focusing on facilitating the work of analysts and developers [11, 55] and developing linguistic taxonomies for privacy content analysis [73, 219]. Existing approaches often aim to automatically recognize sensitive personal information in

unstructured text [140, 196, 235], enabling tools such as TABOO [156] and PrivacyBot [212]. However, these efforts primarily target later stages of the software development lifecycle or focus on functional privacy requirements, leaving a significant gap in addressing privacy concerns within the informal and early-stage context of USs.

To address this gap, this thesis proposes a novel approach for detecting privacy requirements from USs by leveraging Transfer Learning (TL) and NLP techniques. TL enables the adaptation of pre-trained deep learning models to new tasks with minimal additional training, providing a resource-efficient solution for addressing domain-specific challenges [107, 213]. Specifically, the proposed method employs a pre-trained Convolutional Neural Network (CNN) designed for privacy-related text classification [140], enhanced with features derived from a privacy dictionary to capture semantic and using peculiar NLP techniques for syntactic nuances in USs. This combination facilitates the accurate identification of privacy-related elements within Agile requirements specifications.

Through the development and evaluation of this approach, this chapter contributes to the field by providing:

1. A systematic method for detecting privacy requirements from USs using state-of-the-art NLP and TL techniques.

2. Empirical validation of the method on a real-world dataset of 1,680 USs [53], demonstrating its effectiveness in comparison with traditional and deep learning methods.

3. Insights into the role of privacy-specific features in improving classification performance, addressing a critical gap in early-stage privacy requirements engineering.

## 3.2 METHODOLOGY

This section describes the methodology designed for detecting privacy-related disclosures in Agile USs. The approach combines linguistic resources, NLP techniques, and deep learning architectures to identify privacy-related elements in USs effectively.

### 3.2.1 *User Stories as Input*

USs are short, structured natural language requirements used in Agile development. A typical US follows the structure:

`As a` [*role*], `I want to` [*feature*], `so that` [*reason*].

Despite this structured format, identifying privacy disclosures can be challenging due to the variability in terminology and contextual usage of words. To address these issues, the proposed method integrates advanced linguistic and ML techniques.

### 3.2.2 *Lexicon-Based Privacy Features*

The first step involves extracting privacy-related features using a privacy dictionary developed by Vasalou *et al.* [219]. This dictionary categorizes words and phrases into different privacy-related categories, such as *OpenVisible*, allowing the identification of privacy-sensitive language within USs. Figure 3.1 shows an example US with privacy-related words highlighted, while Table 3.1 describes the corresponding privacy category.

**As a site member, I want to access to the Facebook profiles of other members so that I can share my experiences with them.**

Figure 3.1: Example US with words related to the *OpenVisible* privacy category highlighted.

Table 3.1: Privacy category of "access" and "share" [219].

| Category Name | Description | Example Words |
|---|---|---|
| *OpenVisible* | Open and public access to people | port, display, accessible |

### 3.2.3 *NLP-Based Features*

In addition to lexicon-based features, linguistic features are extracted using the spaCy NLP toolkit[1]. These features include parts of speech (POS), syntactic dependencies, and named entities, which help to capture the contextual and structural characteristics of the text. For instance, Table 3.2 shows the POS and dependency information extracted from the US in Figure 3.1.

Table 3.2: Parts-of-speech and dependencies extracted from an example US.

| **Text** | **Part of Speech (POS)** | **Dependency** |
|---|---|---|
| As | SCONJ | prep |
| a | DET | det |
| site | NOUN | compound |
| member | NOUN | pobj |
| I | PRON | nsubj |
| want | VERB | ROOT |
| to | PART | aux |
| access | VERB | xcomp |
| to | ADP | prep |
| the | DET | det |
| Facebook | PROPN | compound |
| profiles | NOUN | pobj |
| of | ADP | prep |
| other | ADJ | amod |
| members | NOUN | pobj |
| so | SCONJ | mark |
| that | SCONJ | mark |
| I | PRON | nsubj |
| can | VERB | aux |
| share | VERB | advcl |
| my | DET | poss |
| experiences | NOUN | dobj |
| with | ADP | prep |
| them | PRON | pobj |

1 https://spacy.io/

### 3.2.4 *Deep Neural Network Architectures*

Two deep learning models are developed to classify USs into privacy-related or non-privacy-related categories:

**CNN for NLP-Based Features ($CNN_{NLP}$)**: This model processes the linguistic features described in Section 3.2.3. Its architecture, shown in Figure 3.2, integrates lexical, syntactic, and semantic features to capture privacy-related patterns in text.

**CNN for Lexicon-Based Features ($CNN_{PW}$)**: This model uses features derived from the privacy dictionary (Section 3.2.2). Figure 3.3 illustrates its architecture, which combines feature embeddings from privacy categories.

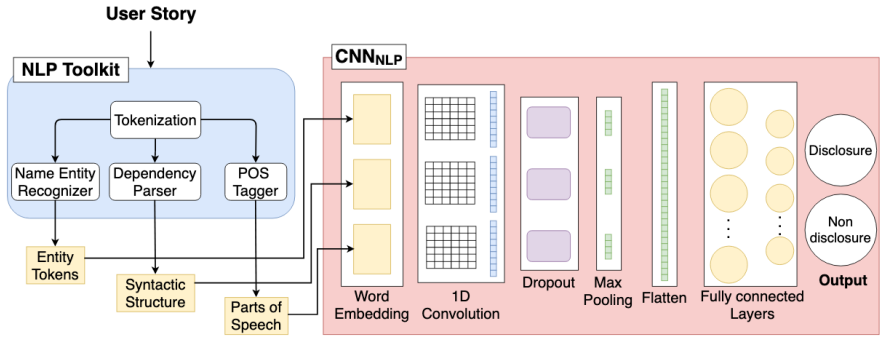


Figure 3.2: The $CNN_{NLP}$ architecture.

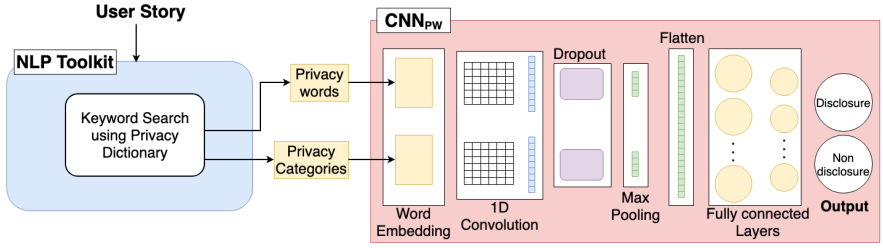


Figure 3.3: The $CNN_{PW}$ architecture.

### 3.2.5 *Transfer Learning for Privacy Disclosure Detection*

To address the scarcity of labeled US datasets, a TL approach is employed. A pre-trained CNN originally developed for detecting private disclosures in Reddit posts [140] is adapted for the privacy detection task. The pre-trained network processes NLP-based features and is fine-tuned using a small labeled dataset of USs. Figure 3.4 shows the architecture of the TL model, $PD_{TL}$, which combines outputs from the pre-trained network with features from the privacy dictionary. This integration enhances the model's ability to capture both linguistic and lexicon-based privacy properties in USs.

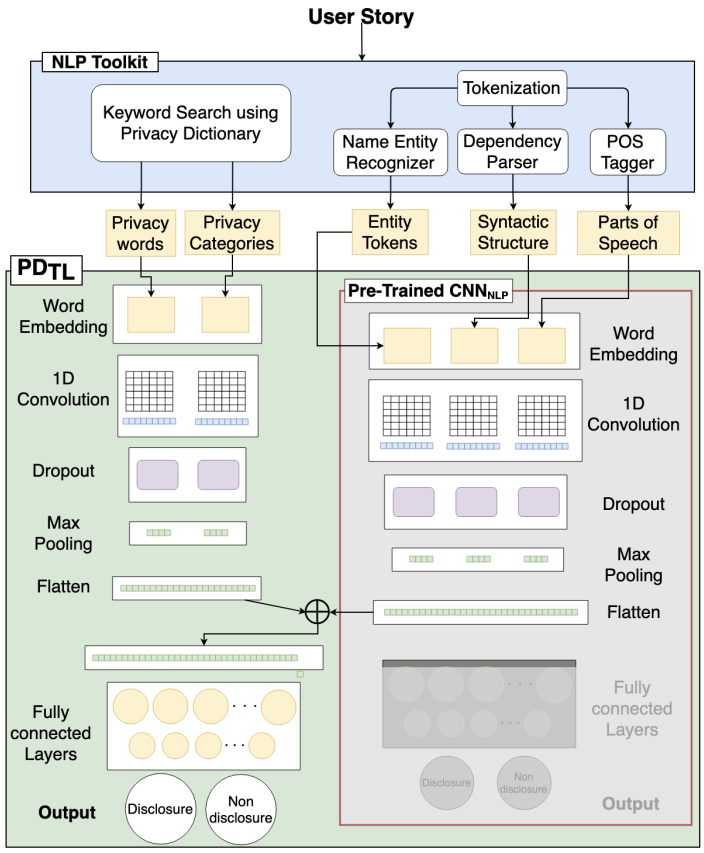


Figure 3.4: The $PD_{TL}$ architecture leveraging transfer learning.

## 3.3 EMPIRICAL STUDY DESIGN

In this section, we outline the design of the empirical study conducted as part of this work. Specifically, we begin by presenting the research questions along with the rationale behind their formulation. Subsequently, we describe the dataset utilized in the analysis, followed by an explanation of the validation methods employed.

The latter part of this section details the evaluation criteria adopted for assessing the predictions made by the ML models. The datasets and scripts used for training the models and reproducing the results are publicly available at https://tinyurl.com/US-privacy.

### 3.3.1 *Research Questions*

This study aims to evaluate advanced methods and technologies for detecting privacy content from USs. As outlined in the introduction, we first perform a preliminary validation to determine:

a) Whether a deep learning method ($CNN_{NLP}$) achieves comparable performance to conventional shallow machine learning methods when utilizing NLP-based features.

b) Whether a deep learning method ($CNN_{PW}$) achieves comparable performance to shallow machine learning methods when using privacy word (PW) features.

Considering that US datasets containing sensitive information are challenging to obtain, we explore TL for privacy disclosure detection, as detailed in Section 3.2.5. TL enables the application of a neural network trained for one task in a given domain to another related task or domain, leveraging shared knowledge. To guide this study, we formulate three research questions:

**RQ$_{Pri}$1** Does $CNN_{NLP}$ perform as effectively as conventional machine learning methods in detecting privacy content when using NLP-based features?

**RQ$_{Pri}$2** Does $CNN_{PW}$ perform as effectively as conventional machine learning methods in detecting privacy content when using PW features?

**RQ$_{Pri}$3** Does $PD_{TL}$, based on transfer learning, yield better predictions than $CNN_{NLP}$ and $CNN_{PW}$?

To address RQ$_{Pri}$1, we trained a $CNN_{NLP}$ using features extracted through NLP techniques. This method predicts whether USs contain privacy-related information or not. CNNs are chosen due to their ability to identify complex patterns even with limited training data. Their effectiveness has been demonstrated in various domains, including natural language processing [132].

For conventional ML methods, we utilized Logistic Regression (LR), Support Vector Machine (SVM), Gaussian Naive Bayes (GNB), k-Nearest Neighbors (KNN), Random Forest (RF), and Decision Tree (DT). When these methods use NLP-based features for training, they are referred to as $LR_{NLP}$, $SVM_{NLP}$, $GNB_{NLP}$, $kNN_{NLP}$, $RF_{NLP}$, and $DT_{NLP}$. The choice of such ML algorithm is deliberate, as these methods are commonly used in software engineering tasks and are well-suited for binary classification problems.

Similarly, RQ$_{Pri}$2 investigates the performance of models trained with PW features. In this case, the models are denoted as $CNN_{PW}$, $LR_{PW}$, $SVM_{PW}$, $GNB_{PW}$, $kNN_{PW}$, $RF_{PW}$, and $DT_{PW}$.

To address RQ$_{Pri}$3, we leverage a TL approach through a CNN model referred to as $PD_{TL}$ (detailed in Section 3.2.5). The expectation is that $PD_{TL}$ will outperform $CNN_{NLP}$ and $CNN_{PW}$ since it combines semantic, syntactic, and keyword-based features to enhance prediction accuracy. In summary, $CNN_{NLP}$, trained on a limited dataset containing privacy-related information, does not exploit PW features, while $CNN_{PW}$ operates on a narrower set of features compared to $PD_{TL}$. By combining these complementary approaches, we aim to improve the detection of privacy-related content in USs, demonstrating the potential of TL for such tasks.

### 3.3.2 *Data Collection*

To train the proposed model for detecting privacy disclosures in USs, it is necessary to use a dataset consisting of USs enriched with labels indicating whether a given US contains privacy disclosures and features that contribute to identifying privacy-related content. Unfortunately, such datasets were not readily available in the literature or online repositories. Consequently, we created a dedicated dataset by identifying and collecting USs, extracting the required features to enable reliable model predictions. To achieve this, we conducted an extensive search to gather a substantial number of USs and identified 22 publicly available datasets, each containing more than 50 USs [53]. The details of the methodology employed to create these datasets are thoroughly documented in [54].

We extracted features from each US in these datasets, such as entities, dependencies, parts of speech, and privacy-related categories and terms. These features serve as independent variables for training the proposed model. In the following, an example is given.
**User Story:** As a data user, I want to have 2017 deletions processed.
**Entities:** ['As', 'a', 'data', 'user', 'PERSON', 'want', 'to', 'have', '2017', 'deletions', 'processed'].
**Dependencies:** ['prep', 'det', 'compound', 'pobj', 'nsubj', 'ROOT', 'aux', 'xcomp', 'nummod', 'dobj', 'acl'].
**Part of speech:** ['SCONJ', 'DET', 'PROPN', 'NOUN', 'PRON', 'VERB', 'PART', 'AUX', 'NUM', 'NOUN', 'VERB'].
**Privacy categories and number of words:** [['PrivateSecret', 1]].
**Privacy words:** ['data'].

Table 3.3 provides an overview of the datasets considered in this study. Each row in the table includes a brief description of the project, the number of USs it contains, the number of privacy terms in those USs, and statistics about the features derived from NLP techniques. Specifically, we processed each US to extract relevant features for model training. The last four columns show the percentage of USs containing: both Privacy Words and Disclosures (*PW&Di*), only Privacy Words (*PW*), only Disclosures (*Di*), and neither (*None*). The first author manually classified the privacy content, and the other authors cross-checked the results.

Table 3.3: Properties of the datasets used for the research.

| # | Description | Size | Privacy Terms | %PW&Di | %PW | %Di | %None |
|---|---|---|---|---|---|---|---|
| 1 | Online platform for delivering transparent information on US governmental spending | 98 | 118 | 0.224 | 0.194 | 0.388 | 0.194 |
| 2 | Electronic land management system for the Loudoun County, Virginia | 58 | 107 | 0.328 | 0.000 | 0.638 | 0.340 |
| 3 | An online platform to support waste recycling | 51 | 86 | 0.176 | 0.137 | 0.137 | 0.549 |
| 4 | Website for create a transparent overview of governmental expenses | 53 | 85 | 0.566 | 0.151 | 0.170 | 0.113 |
| 5 | Platform for obtaining insights from data | 66 | 69 | 0.742 | 0.091 | 0.106 | 0.061 |
| 6 | First version of the Scrum Alliance Website | 97 | 115 | 0.175 | 0.031 | 0.670 | 0.124 |
| 7 | New version of the NSF website: redesign and content discovery | 73 | 115 | 0.041 | 0.000 | 0.740 | 0.219 |
| 8 | App for camp administrators and parents | 55 | 56 | 0.273 | 0.182 | 0.164 | 0.382 |
| 9 | First version of the PlanningPoker.com website | 53 | 53 | 0.170 | 0.057 | 0.623 | 0.151 |
| 10 | Platform to find, share and publish data online | 67 | 63 | 0.552 | 0.134 | 0.104 | 0.209 |
| 11 | Management information system for Duke University | 68 | 132 | 0.206 | 0.191 | 0.206 | 0.397 |
| 12 | Simplified toolbox to enable fast and easy development with Hadoop | 64 | 67 | 0.109 | 0.219 | 0.219 | 0.453 |
| 13 | Research data management portal for the university of Oxford, Reading and Southampton | 102 | 119 | 0.186 | 0.186 | 0.245 | 0.382 |
| 14 | Personal interactive assistant for independent living and active aging | 138 | 126 | 0.036 | 0.065 | 0.413 | 0.486 |
| 15 | Conference registration and management platform | 69 | 106 | 0.116 | 0.430 | 0.739 | 0.101 |
| 16 | Software for machine-actionable data management plans | 83 | 115 | 0.578 | 0.181 | 0.229 | 0.012 |
| 17 | Web-based archiving information system | 57 | 72 | 0.123 | 0.07 | 0.211 | 0.592 |
| 18 | Institutional data repository for the University of Bath | 53 | 89 | 0.660 | 0.038 | 0.226 | 0.075 |
| 19 | Repository for different types of digital content | 100 | 88 | 0.050 | 0.120 | 0.220 | 0.610 |
| 20 | Software for archivists | 100 | 117 | 0.250 | 0.130 | 0.430 | 0.190 |
| 21 | Digital content management system for Cornell University | 115 | 173 | 0.252 | 0.157 | 0.391 | 0.200 |
| 22 | Citizen science platform that allows anyone to help in research tasks | 60 | 82 | 0.050 | 0.067 | 0.400 | 0.483 |

Figure 3.5 presents a breakdown of the dataset, showing the proportions of USs that fall into each category: containing both Privacy Words and Disclosures, only Privacy Words, only Disclosures, or nei-

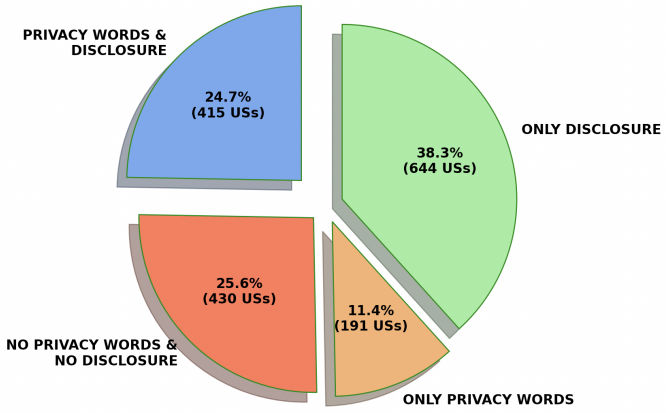


Figure 3.5: Partitions of the dataset for each type of US.

ther. Special attention was given to categories with fewer instances, as insufficient representation could compromise the model's ability to accurately differentiate between them.

The independent variables for this study include features derived from NLP techniques—entities, dependencies, parts of speech, privacy words, and privacy categories. Accuracy and F1-score were selected as the dependent variables to evaluate the model's performance, as explained in the subsequent section.

### 3.3.3 *Evaluation Criteria*

To assess the performance of the proposed models in identifying privacy disclosures, we utilized four widely adopted evaluation metrics for classification tasks [173]: *Accuracy*, *Precision*, *Recall*, and *F1-score*. Each metric provides a different perspective on the model's performance:

***Accuracy***: this metric represents the ratio of correctly predicted observations (*true positive* + *true negative*) to the total number of observations. It is one of the most intuitive performance measures for classification models.

***Precision***: defined as *true positive* / (*true positive* + *false positive*), precision measures the correctness of the responses provided by the model, highlighting its ability to avoid false positives.
***Recall***: this metric, calculated as *true positive* / (*true positive* + *false negative*), evaluates the model's ability to identify all relevant instances, focusing on completeness.
***F1-score***: as the harmonic mean of precision and recall, the F1-score provides a balance between these two metrics. It is particularly useful when a trade-off between precision and recall is important.

These metrics were chosen because they are well-suited for binary classification tasks, where accuracy, precision, and recall are equally important. Moreover, these metrics allowed for direct comparisons between the models implemented in this study and pre-trained models evaluated using the same criteria.

The objective was to analyze how precisely the models identified privacy aspects while understanding their limitations in extracting such information from the test dataset.

Additionally, we evaluated whether the predictions made by the models originated from the same population, ensuring that the observed differences were not due to chance. To this end, we employed the McNemar test, a non-parametric statistical test commonly used for comparing the performance of two models [89, 192]. Non-parametric methods like McNemar's test are preferred in machine learning and deep learning comparisons as they make fewer assumptions about the data.

**McNemar Test:** Given the predictions of two models (Model A and Model B) and the true labels, a contingency table is constructed to record:

1. Cases where both models were correct,
2. Cases where both models were incorrect,
3. Cases where Model A was correct, but Model B was incorrect,
4. Cases where Model B was correct, but Model A was incorrect.

This contingency table is then used to estimate the probability that Model A performs better than Model B as observed in the experiment.

The null hypothesis ($Hn_0$) states that both models are equally accurate in identifying privacy aspects.

*$Hn_0$: All models are equally accurate in identifying privacy aspect.*

For each comparison, a *p-value* of 0.05 was set as the significance threshold. If the *p-value* was below this threshold, the null hypothesis was rejected, indicating that the observed differences were unlikely due to chance. In such cases, it was determined whether one model significantly outperformed the other.

This rigorous evaluation approach ensured the robustness of the proposed models and provided meaningful comparisons to highlight their effectiveness in detecting privacy-related content.

### 3.3.4 *Validation Method*

To evaluate the accuracy and effectiveness of the proposed ML models, it is crucial to assess their performance not only on the training data but also on unseen datasets to understand their generalization capabilities. This is because residual evaluation, which measures errors on the training data, only indicates how well the model performs on familiar data. However, such an evaluation does not provide insight into the model's ability to generalize to new, unseen data, which is often a critical requirement in real-world scenarios.

To address this limitation, we adopted a *k-fold cross-validation*, a widely used technique to assess the model generalizability. The original dataset is partitioned into $k$ subsets (or folds), and the model is trained and validated $k$ times, each time using a different fold as the validation set and the remaining $k-1$ folds as the training set. For our study, we set $k = 5$, thereby dividing the dataset into five equal parts. Furthermore, to ensure the robustness of our evaluation, the 5-fold cross-validation process was repeated 40 times.

The dataset used for training and validation was categorized based on the assumption that determines whether a user story (US) is related to privacy content. Figure 3.5 provides a breakdown of the cardinality of US instances for each type. Using these partitions, the training sets consisted of 664 instances, while the test sets were composed of 166 instances.

Specifically, 50% of the training set consisted of USs containing both privacy words and disclosures (thus, privacy-related requirement), with the remaining 50% distributed among the other three types of USs. For the test sets, half of the instances (83) comprised USs containing both disclosures and privacy words, while the remaining are labeled as non-privacy related.

This validation methodology ensures that the models are rigorously evaluated on multiple data splits, providing a reliable estimate of their predictive performance on unseen data. By leveraging 5-fold cross-validation repeated multiple times, we mitigate the potential biases and variability that might arise from using a single split, thereby ensuring the robustness and reliability of our findings.

## 3.4 ANALYSIS OF THE RESULTS

This section presents and discusses the findings of the empirical study for each research question.

### 3.4.1 *Is $CNN_{NLP}$ accurate at least as conventional machine learning methods to detect privacy content when using NLP-based features?*

To address this research question, we evaluated whether a deep learning method ($CNN_{NLP}$) leveraging NLP-based features performs comparably or better than conventional (shallow) machine learning methods. This represents our first sanity check.

As described in Section 3.3.2, the models were trained and tested with equal proportions of positive and negative samples. Specifically, positive samples correspond to USs containing both Disclosures and Privacy Words. For each fold, 332 positive and 332 negative samples were selected for training, while 83 positive and 83 negative samples were used for testing. Each fold was employed to train and evaluate $CNN_{NLP}$ alongside traditional ML models, including $LR_{NLP}$, $SVM_{NLP}$, $GNB_{NLP}$, $kNN_{NLP}$, $RFC_{NLP}$, and $DT_{NLP}$.

The aggregated results in terms of Accuracy and F1-score are presented in Table 3.4, while Figures 3.6 and 3.7 provide a graphical representation of the results across all runs.

From Table 3.4, it is evident that $CNN_{NLP}$ outperformed the conventional ML methods, achieving both Accuracy and F1-score values greater than 0.7. In contrast, the other methods had values below 0.7. The poorest performance was observed with $SVM_{NLP}$.

Figure 3.6 illustrates the accuracy trends across all runs, highlighting that $CNN_{NLP}$ consistently achieved superior Accuracy values except for four cases. The variations in Accuracy for $CNN_{NLP}$ and other models remained within 10%, with a few exceptions showing a variation of around 20%. Figure 3.7 reveals that $LR_{NLP}$, $kNN_{NLP}$, $DT_{NLP}$, and $RFC_{NLP}$ exhibited regular F1-score trends, with variations within 10%. Conversely, $CNN_{NLP}$, $GNB_{NLP}$, and $SVM_{NLP}$ showed some runs with variations of around 20%, although $CNN_{NLP}$ remained superior in most cases.

Table 3.4: Results achieved with each model to answer $RQ_{Pri}1$ in terms of accuracy and F1-score.

| Model | Accuracy | F1-Score |
|---|---|---|
| **$CNN_{NLP}$** | **0.720** | **0.713** |
| $LR_{NLP}$ | 0.617 | 0.605 |
| $SVM_{NLP}$ | 0.519 | 0.084 |
| $GNB_{NLP}$ | 0.510 | 0.612 |
| $kNN_{NLP}$ | 0.557 | 0.519 |
| $RFC_{NLP}$ | 0.662 | 0.669 |
| $DT_{NLP}$ | 0.609 | 0.611 |

To verify whether the observed performance differences were statistically significant, we applied the McNemar test with the null hypothesis: "there are no differences in the accuracy of the models being compared.". The predictions from $CNN_{NLP}$ were compared against those from each shallow machine learning model ($LR$, $SVM$, $GNB$, $kNN$, $RFC$, and $DT$). For all comparisons, the *p-value* was $<0.001$, allowing rejection of the null hypothesis. These results demonstrate significant differences in accuracy between $CNN_{NLP}$ and the shallow ML methods.

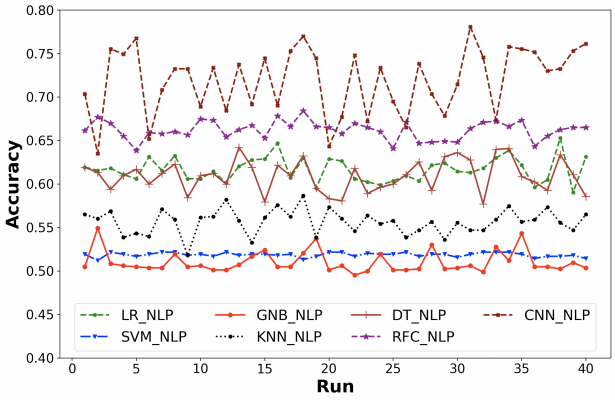


Figure 3.6: Accuracy values for all runs ($RQ_{Pri}1$).

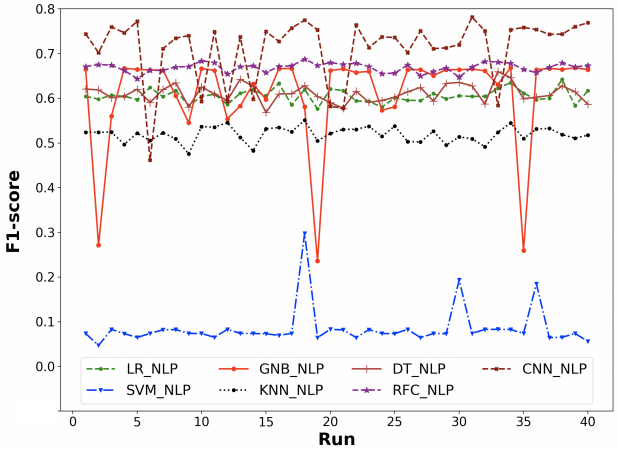


Figure 3.7: F1-score values for all runs ($RQ_{Pri}1$).

In conclusion, the additional effort required to implement CNN is justified by the significant improvement in prediction accuracy.

🔑 ***Thus, we can positively answer our first research question: a deep learning method ($CNN_{NLP}$) provides better predictions than conventional (shallow) machine learning methods.***

### 3.4.2 *Is $CNN_{PW}$ accurate at least as conventional machine learning methods to detect privacy content when using PW features?*

To answer this research question, we evaluated whether $CNN_{PW}$ (a deep learning model leveraging privacy word features) could achieve comparable or better performance than conventional ML methods.

The results of the evaluation, presented in Table 3.5, show that $CNN_{PW}$ achieves strong performance in terms of Accuracy and F1-score, with values of 0.805 and 0.823, respectively. However, $CNN_{PW}$ is outperformed by three conventional ML models: $SVM_{PW}$, $kNN_{PW}$, and $RF_{PW}$, which achieve Accuracy values of 0.828, 0.810, and 0.829, and F1-scores of 0.848, 0.825, and 0.851, respectively. Nevertheless, $CNN_{PW}$ performs better than $LR_{PW}$, $DT_{PW}$, and $GNB_{PW}$ in both.

Table 3.5: Results achieved with each model to answer $RQ_{Pri}2$, in terms of accuracy and F1-score.

| **Model** | **Accuracy** | **F1-Score** |
|---|---|---|
| $CNN_{PW}$ | 0.805 | 0.823 |
| $LR_{PW}$ | 0.801 | 0.819 |
| $SVM_{PW}$ | **0.828** | **0.848** |
| $GNB_{PW}$ | 0.584 | 0.343 |
| $kNN_{PW}$ | 0.810 | 0.825 |
| $RF_{PW}$ | **0.829** | **0.851** |
| $DT_{PW}$ | 0.805 | 0.819 |

Figures 3.8 and 3.9 depict the accuracy and F1-score across all runs. These plots highlight that $CNN_{PW}$ has consistent performance, although it falls short when compared to ML models listed above.

To determine whether these performance differences are statistically significant, we conducted the McNemar test. Specifically, we compared $CNN_{PW}$ with each of the conventional machine learning models ($LR_{PW}$, $SVM_{PW}$, $GNB_{PW}$, $kNN_{PW}$, $RF_{PW}$, and $DT_{PW}$). For all comparisons, the McNemar test yielded a *p-value* $<0.001$, allowing rejection of the null hypothesis.

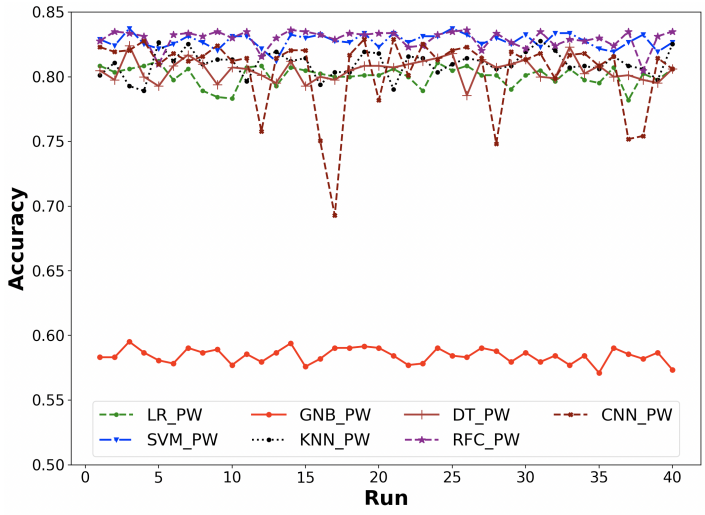


Figure 3.8: Accuracy values of all the runs (to answer RQ$_{Pri}$2).

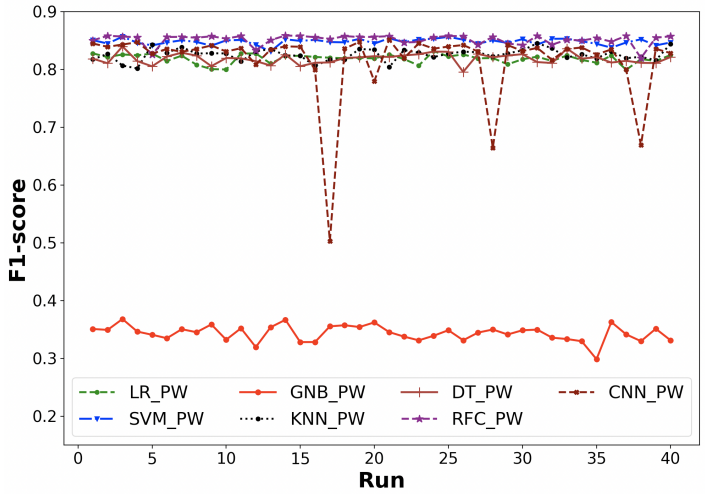


Figure 3.9: F1-score values of all the runs (to answer RQ$_{Pri}$2).

These results confirm that there are statistically significant differences in performance between $CNN_{PW}$ and the other models.

Thus, $CNN_{PW}$ performs better than three conventional machine learning models ($LR_{PW}$, $DT_{PW}$, and $GNB_{PW}$) but worse than the other three ($SVM_{PW}$, $kNN_{PW}$, and $RF_{PW}$) when using PW features.

Table 3.6: Results achieved with each model to answer $RQ_{Pri}3$, in terms of accuracy and F1-score.

| Model | Accuracy | F1-Score |
|---|---|---|
| $CNN_{NLP}$ | 0.720 | 0.713 |
| $CNN_{PW}$ | 0.805 | 0.823 |
| **$PD_{TL}$** | **0.937** | **0.937** |

🔑 ***Therefore, we cannot positively answer our second research question, as the deep learning method $CNN_{PW}$ is not as accurate as all the considered conventional machine learning methods when using PW features.***

Interestingly, this result aligns with findings from previous studies (e.g., [161]) that demonstrate how deep learning models can sometimes underperform in scenarios with limited training data. This underscores the motivation of employing Transfer Learning.

For completeness, it is worth noting that shallow ML models trained with PW features generally outperformed their counterparts trained with NLP-based features (see Tables 3.4 and 3.5). The McNemar test further revealed that these differences are statistically significant, indicating that conventional methods benefit from training with a smaller, well-defined feature set.

### 3.4.3 *Are predictions obtained with $PD_{TL}$ better than those achieved with $CNN_{NLP}$ and $CNN_{PW}$?*

The goal of this research question is to evaluate the effectiveness of applying $PD_{TL}$ for detecting privacy content in USs, in comparison to the deep learning models $CNN_{NLP}$ and $CNN_{PW}$.

Table 3.6 presents the results achieved by $PD_{TL}$, $CNN_{NLP}$, and $CNN_{PW}$ in terms of Accuracy and F1-score. $PD_{TL}$ outperforms both $CNN_{NLP}$ and $CNN_{PW}$ significantly, achieving values greater than 0.93 for both metrics. This represents an improvement of more than 10% over the best results achieved by $CNN_{PW}$.

Figures 3.10 and 3.11 graphically depict these results, illustrating the consistent superior performance of $PD_{TL}$ across all runs and demonstrating that $PD_{TL}$ achieves higher prediction consistency compared to the other models. The accuracy and F1-score values for $PD_{TL}$ exhibit less variation across different runs, while both $CNN_{NLP}$ and $CNN_{PW}$ show higher variability in their predictions.

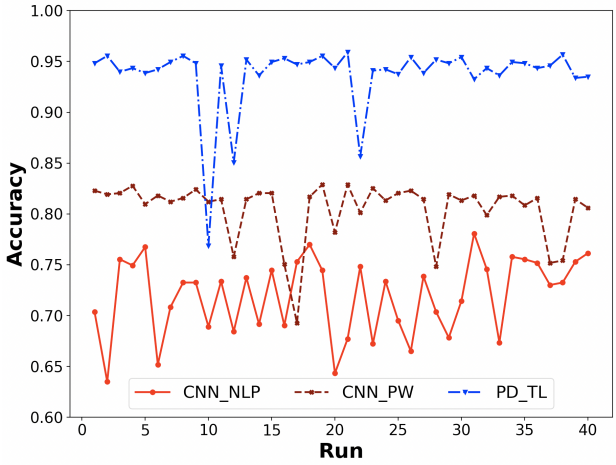


Figure 3.10: Accuracy values of all the runs (to answer RQ$_{Pri}$3).

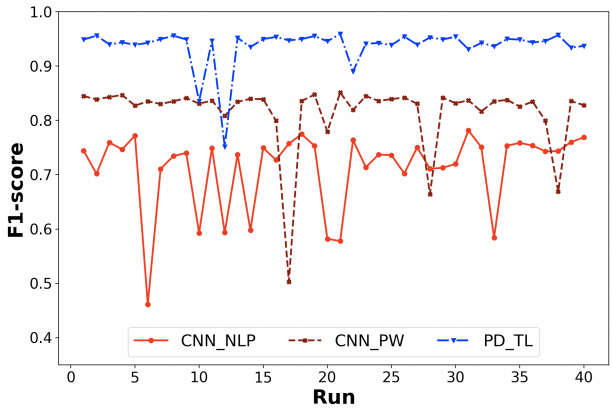


Figure 3.11: F1-score values of all the runs (to answer RQ$_{Pri}$3).

Comparing $PD_{TL}$ and the other models ($CNN_{NLP}$ and $CNN_{PW}$) by the mean of McNemar test, the former yielded a *p-value* $< 0.001$ in all the cases, allowing us to reject the null hypothesis that the differences are due to chance. Based on these findings, we can conclude that Transfer Learning not only improves prediction accuracy and F1-score but also ensures more stable and consistent performance across different runs. This demonstrates the feasibility and effectiveness of Transfer Learning for privacy content detection in USs.

🔑 ***Thus, we can positively answer our third research question: predictions obtained with $PD_{TL}$ are better than those achieved with $CNN_{NLP}$ and $CNN_{PW}$.***

## 3.5 FINDINGS FOR RESEARCHERS AND PRACTITIONERS

The analysis conducted to address our research questions provides significant insights and implications for both researchers and practitioners regarding the applicability and relevance of our findings. We structure the discussion based on the major contributions achieved.

**On the use of a tool to predict privacy content.** We have introduced an approach and a tool designed to automatically predict privacy content from user stories—a problem that has not been addressed previously. This approach combines NLP and Transfer Learning strategies, offering a novel and effective way to address privacy content detection. This contribution should motivate researchers and practitioners in the field of software engineering to explore the potential benefits of automating privacy detection processes.

👉 *Implication 1. Practitioners are now equipped with a method and tool that significantly reduce the effort and cost associated with identifying privacy requirements during the early design phase. Further user studies should involve practitioners to validate and promote the suggested approach and tool in real-world scenarios.*

**On the use of deep learning methods.** As anticipated, the experimental results demonstrate that employing NLP-based CNNs enhances the accuracy of privacy requirement predictions compared to conventional (shallow) ML techniques. However, the findings highlight the critical importance of the strategies used to train the models.

For example, the $RQ_{Pri}2$ analysis did not reveal a consistent advantage of deep learning methods over shallow ML methods, aligning with findings in other studies [161].

☛ *Implication 2. Researchers should prioritize empirical studies across diverse datasets to identify effective strategies for training NLP-based prediction models for privacy requirement detection in agile contexts.*

**On the use of privacy words.** Our analysis highlighted the role of privacy words in enhancing the performance of the ML methods employed. The results from $RQ_{Pri}2$ (compared to $RQ_{Pri}1$) demonstrated a significant improvement when privacy words were used.

☛ *Implication 3. The research community should explore the influence of domain-specific data on the effectiveness of simpler and cost-effective methods compared to more sophisticated but resource-intensive techniques.*

**On the use of Transfer Learning.** The most notable outcome of our analysis is the application of Transfer Learning, which improved the performance of the built NLP-based CNN prediction models by approximately 10% in terms of Accuracy and F1-score (as shown in $RQ_{Pri}3$ results). This further validates the advantages of this emergent strategy, which facilitates the reuse of systems developed for one task to build models for different yet related tasks [107, 108, 213].

☛ *Implication 4. Researchers can leverage Transfer Learning to train NLP-based prediction models, not only for privacy detection but also for security requirements in agile development contexts.*

## 3.6 THREATS TO VALIDITY

This section outlines the main threats to validity associated with the study, discusses their potential effects, and describes the measures taken to mitigate them.

**Construct Validity:** This pertains to ensuring that the measurement methods accurately correspond to the constructs being evaluated. In this study, Scikit-learn library methods were used to evaluate performance metrics, specifically the `f1_score`[2] for F1-score and the

2 https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

`accuracy_score`[3] for Accuracy. These metrics were selected due to their established usage in prior studies on privacy disclosure detection, allowing for meaningful comparisons with previous findings. While reliance on a single library may pose a threat to construct validity, the choice of Scikit-learn was necessary to ensure compatibility between traditional machine learning models and deep learning methods implemented in Keras.

Another important threat to construct validity in this study is related to the labeling of NFRs in the datasets used for training and evaluation. When using NLP models to classify text, it is assumed that the ground truth labels (e.g., Privacy, Security, Fairness) are correctly assigned. However, human annotation is inherently prone to error. For instance, in this study, a requirement tagged as *privacy* may have been incorrectly classified due to ambiguous wording, or because annotators applied different interpretations. Additionally, some words that appear privacy-related may not actually refer to privacy in the given context, leading to potential noise in the dataset. Perhaps, there remains a residual risk that some USs were misclassified. This could impact performance metrics such as accuracy, precision, and recall, so future work could further validate the dataset to enhance the robustness of privacy classification.

**Internal Validity:** Internal validity concerns the control of extraneous variables and external factors that may influence the study's results. In this work, exploring the applicability of transfer learning for detecting privacy aspects, it was assumed that the models employed were compatible as they were developed using the same technology (i.e., Keras). Future studies could investigate the integration of models or neural networks developed through different frameworks, such as Keras and PyTorch[4], in transfer learning experiments. Additionally, while causality may pose a potential threat, statistically significant correlations were observed across measures obtained using different methods, supporting the assumption that these relationships are underpinned by robust causal links.

3 https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html

4 https://pytorch.org/

**External Validity:** External validity refers to the generalizability and repeatability of the study's findings. Although the proposed approach is implemented in Python, the statistical models and methods used are programming language-agnostic and can be replicated in other programming environments, provided suitable frameworks are available. To promote reproducibility, all tools, scripts, and data used in this study are made publicly available as mentioned earlier.

**Conclusion Validity:** This aspect assesses the reasonableness of the conclusions drawn from the research or experimental findings. While the number of observations made for statistical tests may not be extensive, the necessary hypotheses were thoroughly validated. Therefore, the relationships between the data and the results are considered robust and the conclusions drawn from the analysis are deemed reasonable.

# 4

# REFAIR: TOWARD A CONTEXT-AWARE RECOMMENDER FOR FAIRNESS REQUIREMENTS ENGINEERING

## 4.1 INTRODUCTION

In today's software production, data-driven machine learning (ML) algorithms are more and more employed to support decision-making activities performed by individuals and companies [238], other than to automate repetitive tasks, reducing human's workload [186].

Successful applications have been showcased in multiple domains [64] like loan management [163], hiring decisions [149], healthcare [2], biology [211], and more.

Despite the benefits brought by ML, researchers have been reporting on the implications that those algorithms may have on ethics and fairness [142]: the reliance on historical data may lead an ML algorithm to gather biased knowledge about the relations ruling a phenomenon, which might lead to unfair predictions and recommendations that, in turn, might reiterate discrimination and injustice [12]. As such, the definition of methods and tools able to reduce risks due to ethical concerns represent a key challenge [12, 43].

Recently, the *Software Engineering for Artificial Intelligence* (SE4AI) research community has been actively working on this matter, arguing the need for novel engineering processes to treat fairness throughout the software lifecycle. However, operationalizing this need is complex because of the intrinsic nature of fairness.

On the one hand, fairness strictly depends on the application domain and the specific task an ML system is designed for [141, 171], e.g., a feature may be sensitive in one context and not in another. On the other hand, fairness represents a multi-faceted aspect, and, indeed, multiple definitions targeting various perspectives of software fairness have been proposed in literature [222].

At the current stage, most of the existing approaches focus on avoiding discrimination from data. For instance, Chakraborty et al. [36] proposed Fair-SMOTE, an oversampling algorithm able to rebalance training data according to sensitive attribute groups. On a similar note, other researchers attempted to analyze the best data preprocessing actions to keep fairness under control [17], how to diversify data to reduce fairness concerns [152], and optimize training data to balance fairness and accuracy [42, 81]. At the same time, automated fairness testing procedures have been defined [70].

Recognizing these advances, Soremekun et al. [205] pointed out the need for novel requirements engineering techniques that may let practitioners be aware of sensitive features since the project inception. Those instruments may have a fundamental impact on practice: being able to provide early recommendations on sensitive features, they may inform the entire ML engineering lifecycle, possibly making all the involved stakeholders aware of the most suitable bias mitigation strategies to put in place to reduce risks due to unfairness. In addition, the outcome of such a recommender may complement existing bias mitigation approaches, empowering the whole ML pipeline by making it more fairness-aware.

In this work, we perform the first step toward this objective and propose ReFair, an automated requirements engineering framework that employs natural language processing (NLP) and word embedding techniques to classify sensitive features from User Stories (USs).[1] We design ReFair to be *context-aware*. As such, it can classify application domains and ML tasks to be implemented before recommending the sensitive features to consider, hence addressing the needs brought by the intrinsic nature of software fairness.

To experiment with ReFair, we create a *synthetic* dataset of 12,401 ML-related USs pertaining to 34 different application domains. The results of our study showcase the capabilities of ReFair, which can (1) classify application domain and ML tasks within the USs with an F1-score of 97% and 90%, respectively, and (2) recommend sensitive features with high precision.

1 The framework analyzes US titles rather than the entire structure of a US; yet, for the sake of readability, we use the term "User Story" throughout the manuscript.

To sum up, our research provides three key contributions:

1. ReFair, a novel context-aware automated framework to support fairness requirements engineering;
2. the empirical validation of ReFair, which showcase the capabilities of our framework;
3. a publicly available replication package [184] which includes
   (a) the implementation of ReFair,
   (b) the dataset and scripts used to assess the framework,
   (c) a technical report discussing the additional analyses conducted to assess its capabilities.

## 4.2 BUILDING A DATASET OF USER STORIES

One of our work's challenges was the identification of a dataset of ML-enabled system's requirements. It should have reported *domain-specific* requirements of ML-enabled systems, it should have been *diverse* enough to investigate software fairness in various domains, *large* enough to experiment with our approach, and *generic* enough not to be explicitly tailored on fairness analysis.

Unfortunately, the current literature does not offer any off-the-shelf solution. Hence, we proceeded with the creation of a *synthethic* dataset, which we built by considering (i) the contemporary requirements engineering processes [85], (ii) the knowledge on the domains where fairness impacts ML solutions [205], and (iii) the trustworthiness of the generation process [230].

Figure 4.1 overviews the generation process.

### 4.2.1 *Requirements Format Selection*

Among the available standards [25, 51, 201], we focused on *User Stories* (USs). These represent a widely adopted instrument to describe features of a software system from the perspective of the users that will interact with the system being developed [85].

Requirements engineering processes of ML-enabled systems are typically performed by means of USs [223]. USs enclose three main elements [51]: (1) the *actor*, who is the main user interested in performing a specific task; (2) the *action*, i.e., the activity to perform using the system under development; and (3) the *benefits*, i.e., the advantages the actor (and the application domain) has from acting on the environment through the feature. Among the main advantages of using USs, we identified the possibility to enclose as actions the specific ML tasks that a software system should enable [122]. In addition, the actor and benefits may typically enclose information on the domain where the system should act, hence possibly providing insights into the context-dependent fairness aspects to consider.

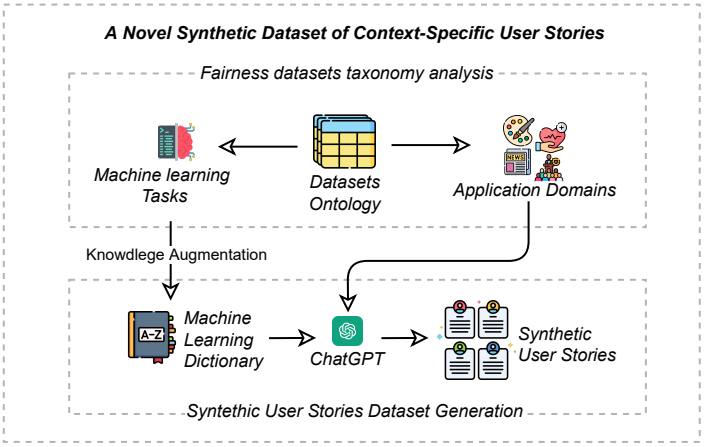


Figure 4.1: An overview of the USs Dataset Generation Process.

### 4.2.2 *A Taxonomy of Fairness-Related Application Domains and ML Tasks*

The second step toward the definition of our synthetic dataset consisted of mapping the existing knowledge concerned with the application domains and ML tasks for which software fairness concerns arise. This step was key in our case to inform the US generation process, i.e., without such a systematic mapping, we could not know for which domains and ML tasks USs should have been generated.

In particular, this was a two-step process that included (1) the manipulation of an existing ontology describing fairness-critical application domains and ML tasks [64]; and (2) the augmentation of such an ontology with the specific ML techniques for the various ML tasks.

**Mapping the Existing Knowledge.** We first exploited the OWL ontology developed by Fabris et al. [64], which maps over 250 fairness-related datasets onto the application domains and ML tasks for which they were used and the corresponding sensitive features representing possible causes of unfairness. To the best of our knowledge, this represents the most updated resource available that describes software fairness in different contexts. The ontology was ideal for our case, as it provides information on (i) the domains we should have considered while generating the synthetic dataset of USs and (ii) the ML tasks that might produce fairness concerns and that we should have further analyzed during the generation process.

To make the ontology functional to our purposes, the first author of this article—a SE research assistant with two years of expertise in ML, SE4AI, and ethical AI—manually converted it into a database reporting (1) the application domains and ML tasks classified by Fabris et al. [64]; and (2) the sensitive features impacting fairness within each of the application domains and each of the ML tasks. Overall, the database included 34 application domains and 25 ML tasks, along with the sensitive features that impact them. We made the converted ontology available in our online appendix [184].

**Augmenting the Existing Knowledge.** While the original ontology provided us with an extensive amount of application domains, the set of ML tasks was quite restrictive. The ontology classified the tasks based on a high-level categorization, e. g.ML classification or regression. However, those tasks could be implemented using a variety of models and algorithms. Building a synthetic dataset solely relying on such a high-level classification might have negatively impacted the conclusion validity of our study. Indeed, USs may be defined by specifying either the problem or the solution of the ML tasks to be developed [223], for instance they may either indicate the general classification task to be performed (e.g. classification) or the specific technique that will be used to implement a requirement (e.g. a *Naive Bayes* classifier).

To tackle this problem, we performed a data augmentation process aiming at enlarging the set of ML tasks considered by the original ontology. We exploited the AI dictionary proposed by Duran Silva et al. [61]: this is a collection of 599 specific words widely used in different artificial intelligence areas, such as machine learning and natural language processing. This dictionary was built using advanced language models and various large knowledge datasets such as arXiv, Dpedia, Wikipedia, and Scopus. Experts from several universities have validated the final collection of keywords. Among the keywords of the dictionary, 457 of them relate to ML or natural language processing techniques, i. e.they report about learning algorithms or models. The rationale behind the use of the dictionary was that of exploiting this knowledge to link finer-grained techniques to the higher-level tasks reported by the original ontology, hence creating a comprehensive taxonomy that reports, for each application domain, both high-level ML tasks and low-level ML techniques that might lead to fairness-related concerns.

The mapping was manually performed by the first author, who is referred as "the inspector" in the following. For each of the 457 relevant ML keywords of the dictionary by Duran Silva et al. [61], the inspector (1) verified that the keyword was actually related to a ML or NLP technique, i. e.the keyword matched an algorithm or model—otherwise, the keyword was discarded; and (2) mapped the technique onto one of the 25 higher-level ML tasks. When the inspector was unfamiliar with the specific technique considered, online material, books, or the other authors could be consulted to understand how the mapping should have been performed, i. e.which higher-level task would have better suited the technique. The inspector did not find cases where the mapping could not be done: techniques reported in the dictionary could be successfully mapped, increasing the confidence in the choice of relying on the work by Silva et al. [61] to augment the original ontology by Fabris et al. [64].

The other authors then verified the consistency of the mapping. The disagreements cases were discussed and solved before proceeding to the next stages. As an outcome of this stage, we could rely on a comprehensive, multi-level taxonomy that reported the application domains where fairness concerns arise along with the high-level

ML tasks and low-level techniques that may possibly induce the emergence of fairness issues. In addition, it is worth remarking that—exploiting the knowledge collected through the original ontology—those pieces of information are *directly* mapped onto the specific sensitive features that may cause fairness issues: in other terms, by design, our augmented taxonomy can be used as a basis to create USs that actually provide insights into the application domains and ML techniques that may typically lead ML engineers to deal with sensitive features.

Figure 4.2 reports a snippet of the taxonomy, showcasing examples of mapping between ML tasks and sensitive features and between application domains and sensitive features. The whole taxonomy is available in our online appendix [184].

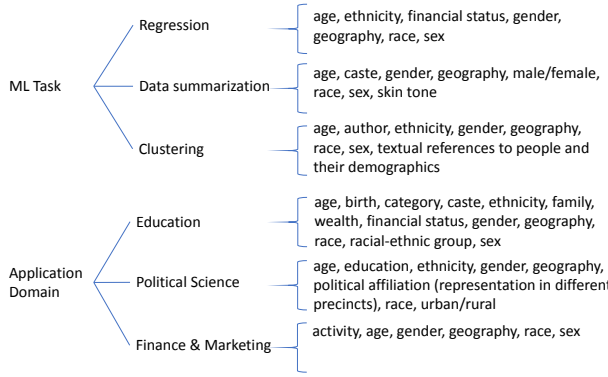


Figure 4.2: Snippet of the augmented taxonomy built.

### 4.2.3 *Synthetic Generation of User Stories*

The taxonomy represented the input of the final step of the synthetic dataset generation process. To generate USs as realistic as possible, we exploited the capabilities of large language models (LLMs) [22] and, in particular, of ChatGPT.[2] This is one of the most powerful LLMs currently available and is based on the GPT-3.5 architecture.[3]

2 Link to ChatGPT: https://openai.com/blog/chatgpt.
3 More on GPT 3.5: https://platform.openai.com/docs/models/gpt-3-5.

A key challenge was represented by the so-called *prompt engineering* [240], i.e., the definition of the most suitable query that would have allowed ChatGPT to properly process the input and output USs that would have effectively mimicked the actions done by a software engineer. We experimented with multiple prompts, finally coming up with the one whose structure is reported below.

**Prompt employed to generate USs.**
Considering the following:
*[High-level machine learning task]*
OR *[Low-level machine learning technique]*
in the field of *[machine learning]*
OR *[natural language processing]*.
Can you provide me with specific user stories
for the following application domains?
*[List of Relevant Application Domains]*

The prompt was employed to systematically query ChatGPT and generate a number of domain-specific USs equals to the number of ML tasks/techniques that may induce fairness-related issues, according to the taxonomy built in the previous step. Such a procedure aimed at emphasizing the *context-dependent* nature of ML fairness, putting a strong focus on domain and machine learning task specificity that serve as the foundation of our work. The generation process was *supervised*, i.e., we did not blindly rely on the USs generated by ChatGPT as these might have been unrealistic or erroneous, affecting the overall reliability and representativeness of the dataset. Every time a new US was generated, the first author manually verified its consistency and degree of realism, discarding low-quality USs. As a result of this manual validation, about 3,000 USs targeting 81 of the ML tasks originally considered in the augmented taxonomy were removed, as they were deemed too specific for generating USs with a structure and expressiveness close to the ones produced in real-world development environments.

Overall, the dataset generation process required around 120 person/hours and produced 12,401 synthetic USs related to 34 different application domains. Upon completion of the generation process, the second author double-checked the operations conducted by the first author on a statistically significant sample of 375 synthetic USs

(confidence level=95%, margin of error=5% - min 373 instances) to (i) further ensure the reliability of the produced dataset and (ii) to assess its suitability in the subsequent experimental phases. More specifically, the second author conducted a qualitative evaluation on the statistically significant sample of synthetic USs which involved a review against predefined criteria, including clarity, completeness, and relevance to the anticipated experimental conditions. The analysis revealed no inconsistencies. Our online appendix provides access to the dataset, other than to additional reports and examples on the dataset generation process and its validation [184].

The prompt provided to ChatGPT may generate synthetic USs having different levels of granularity, i.e., they may specify the solution or the problem of the ML tasks to be developed. The motivation behind the choice of the granularity of the generated USs comes from existing literature in the requirements engineering domain [122] that showed that the granularity of USs may largely vary in practice.

The original Cohn's user story template [51] indeed described a template to write user stories but it did not provide stringent constraints on how to write them. As a consequence, any development team is free to decide on the level of granularity based on multiple factors like the knowledge of the domain, the information made available by the client, etc. [128]. Our synthetic dataset generation procedure embedded these considerations and attempted to simulate the behavior of the largest population of requirements engineers. As such, we kept the generation as broad as possible, foreseeing the possibility of having a diverse set of USs.

To showcase the levels of granularity of the USs in our dataset, let consider the two examples depicted in Figures 4.3 and 4.4. In both cases, the US pattern defined by Cohn [51] is fully met. The former describes a solution-oriented US concerned with the application domain named *"Finance and Marketing"*, where a rather specific ML task is specified, i.e., *"Nearest Neighbor Search"*. The latter, instead, refrains from specifying a ML task and revolves around the problem of *"terminology extraction"*, which can be later implemented using different methods. In this instance, the "what" conveyed is the user's desire to *"extract the key terminology from a large corpus of academic"*, maintaining a high-level abstraction of the ML task.


Figure 4.3: Example of solution-oriented US generated.

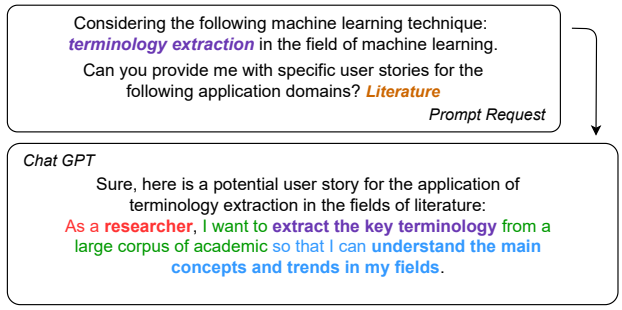


Figure 4.4: Example of problem-oriented US generated.

### 4.2.4 *Synthetic Dataset Validation*

The *supervised* dataset generation process ensured that the synthetic USs closely resembled the ones that a software engineer would produce, as the internal assessments suggested. This validation instilled a high level of confidence in the dataset's reliability. Nevertheless, to further safeguard against potential subjectivity in the internal inspection, we undertook an additional and comprehensive validation of the synthetic dataset. In particular, we let practitioners assess the USs's quality in our dataset. First, we extracted a statistically significant different sample than the one used for the internal inspection of 375 synthetic USs (confidence level=95%, margin of error=5% - min 373 instances). Second, we randomly split the 375 USs into 75 groups, each containing five USs. Third, we designed 75 online surveys - one

for each USs group - to (1) present each of the five USs and inquire the respondents about *comprehensibility*, i.e., the degree to which the US is understandable, *realism*, i.e., the degree to which the US is written as a real user story, and *actionability*, i.e., the degree to which the US can be used to drive the development of a ML-enabled project (the respondents judged these properties for each proposed US through a 5-point Likert scale [14]); and (2) ask respondents to provide feedback on how to improve the US deemed unrealistic. To avoid respondents being biased, in either positive or negative fashion, we did not reveal the synthetic nature of the USs under evaluation. We administered the survey through PROLIFIC[4], following the guidelines by Reid et al. [187] to prevent invalid responses, and applying quality assessment to discard unreliable responses. We cross-validated two reports for each USs group, thus involving 150 practitioners with experience in software design and ML. The results of the surveys corroborated those of the internal validation: the involved practitioners (i) gave the USs very high scores for all indicators, achieving at least $\approx$4 on the Likert scale; (ii) provided recommendations on how 46 unique USs (12%) could be improved. These recommendations were minor and addressed ambiguous terminology affecting the readability of the USs, not altering their core structure or intent. As such, we did not require to propagate the changes to the other USs in the sample nor to the entire dataset. The external validation is more detailed in the technical report released in our online appendix [184].

## 4.3 THE REFAIR FRAMEWORK

The key idea behind REFAIR is to analyze USs of ML-enabled systems with the aim of (1) *classifying* the application domain of the system being developed; (2) *classifying* the ML task(s) that will be employed to implement the US; and (3) *mapping* those pieces of information onto the specific sensitive features to account when working under the classified application domain and ML task.

As such, REFAIR supports the requirements engineer by providing recommendations that may inform the follow-up development

4 The PROLIFIC administration platform: https://www.prolific.co/

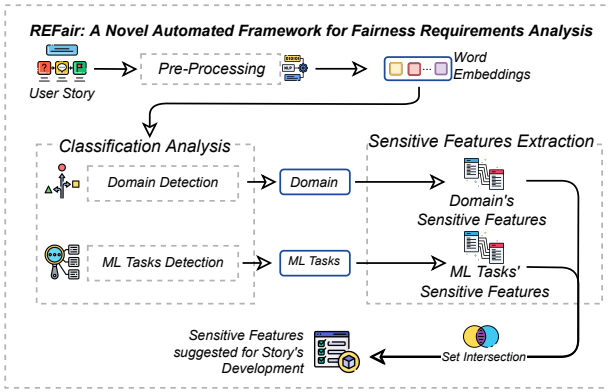


Figure 4.5: REFAIR: An overview of the proposed approach.

activities of the potential fairness concerns to take into account. We are aware that other external factors, e.g., laws and regulations, may influence the identification of sensitive features. On the one hand, REFAIR is designed to work with software engineering artifacts, i.e., User Stories, rather than with elements that may be hardly extracted because of their tight relation with the specific customs or regulations of the society where a system is being developed. On the other hand, our framework does not aim at replacing the requirements engineer but provide insights that may be further elaborated. From a technical standpoint, REFAIR exploits the *base ontology* built in Section 4.2 to learn how to detect sensitive features based on application domains and ML tasks, and is fed with the set of synthetic USs coming from the dataset generation process.

We made REFAIR working with the base ontology rather than with the augmented taxonomy as we aimed at designing a *generalized* framework that could not limit itself to the analysis of the currently existing ML techniques, but that would rather allow the classification of a set of general high-level tasks that can be adapted to multiple application domains, hence recommending sensitive features independently from the specific ML techniques that engineers will use. Such a generality has an additional implication: the best ML technique to use in a given context may result from experimental analyses performed after the requirements engineering phase. Relying on a

higher-level task classification can better inform how such investigations should be performed. Hence, we argue that such a design choice better fits the ML engineers' needs. On the contrary, feeding and experimenting the framework with the USs coming from the augmented taxonomy was key to understand the robustness of the framework. Those USs were based on different levels of granularity, hence simulating multiple conditions arising in reality which might challenge the capabilities of our framework.

Figure 4.5 overviews the main steps of the proposed approach. A US represents the input of ReFair. This is preprocessed with the aim of producing a word embedding representation of the elements of the US. Those embeddings will feed two different ML models:

the first will be responsible for classifying the application domain of the US, while the second will classify the most likely ML task(s) that may be employed to implement the US. The outcomes of these two models will then be used to map the classified application domain and ML tasks to the corresponding sensitive features. These will be finally presented to the user. The next sections provide more details on each of the steps of our approach.

### 4.3.1 *User Story Preprocessing*

The first step of ReFair allows to produce a $N$-dimensional space representation of the input US [113]: in turn, such a representation enables the extraction of features out of the text that natural language models can use for classification purposes. In other terms, this step allows ReFair to transform the text contained within a US into a real-valued vector that can be used in the following steps. Our framework supports multiple word embedding techniques such as TF-IDF [189], BERT [57], Word2Vec [146], FastText [19], and GloVe [170]. The assessment reported in Section 4.4 aimed at experimenting with those techniques and identifying the best one.

### 4.3.2 *Classification Analysis*

The word embeddings are then taken as input by the classification analysis modules of ReFair. This comprises two main components:

**Application Domain Classification.** This component is responsible for classifying the most likely application domain of the US among the 34 domains available in the ontology.

We have modeled the domain detection problem as a multi-class classification task [94], where (1) the features are represented by the real-valued vector of the word embeddings and (2) the classification labels correspond to the application domains of the augmented taxonomy. Our framework supports 25 ML algorithms that make different assumptions on the underlying data as well as have different advantages and drawbacks in terms of execution speed and overfitting. For instance, ReFair provides users with the possibility to run probabilistic algorithms, e. g.Gaussian Naive-Bayes, entropy-based classifiers, e. g.Random Forest, semi-supervised approach, e. g.Label Propagation, and others. Our online appendix [184] reports the complete list of ML techniques supported.

**Machine Learning Tasks Classification.** This is responsible for classifying the ML tasks likely to be employed when implementing the US. We have modeled the problem as a multi-label classification task [215], as a US may be operationalized using multiple ML techniques. For instance, the term *"artificial neural network"* may refer to tasks of regression, classification, clustering, and more. As such, we designed the framework to be *conservative* enough and identify all the potential ML tasks that may lead to unfairness. From a practical perspective, this choice may allow the users to receive a larger set of sensitive features, hence favoring recall over precision: this was done on purpose, as we preferred to provide users with actionable feedback that might be later interpreted rather than with a more restrictive set of sensitive features that may have overlooked some relevant pieces of information. The implications of these design choices are later analyzed as part of the empirical study. In this case, the features are represented by the real-valued vector of the word embeddings, while the classification labels consist of the high-level ML tasks of the augmented taxonomy.

As for the actual classification, ReFair supports established multi-label techniques such as Binary Relevance (BR) [126], Classifier Chain (CC) [185], and Label Powerset (LP) [209]. We rigorously tested these techniques with popular ML algorithms such as Logistic Regression (LR), Random Forest (RF), Gaussian Naive Bayes (GNB), Linear Support Vector Classification (LSVC), K-Nearest Neighbors (KNN), and Decision Tree (DT). We did not consider the analysis of more advanced ML solutions, e. g.Deep Learning (DL) neural networks, as (1) shallow ML classifiers are more interpretable and explainable, possibly providing ReFair with an additional, relevant feature that would increase its practical usability; (2) DL models may not be required if the performance of shallow ML classifiers are already high.

### 4.3.3 *Sensitive Features Recommendation*

The application domain and ML tasks classified in the previous step are finally used to recommend sensitive features. ReFair exploits the base ontology [64] to identify the sensitive features connected to both the application domain and ML tasks concerned with the the classified domain. The intersection of those sensitive features represents the final outcome of the framework, i. e.the outcome comprises the set of sensitive features relevant when jointly considering the application domain and the learning tasks.

### 4.3.4 *Prototypical Implementation*

We released the source code of the prototypical implementation of ReFair in our online appendix [184]. For the sake of understandability, Figure 4.6 reports a running example of our framework, which shows how it could recommend sensitive features for the US generated in Figure 4.3. The example shows a successful classification from the empirical study discussed later, in partucular the results obtained when configuring the framework with (1) the XGBoost classifier for the domain classification and (2) Linear Support Vector Machine for the ML tasks classification. The approach cor-

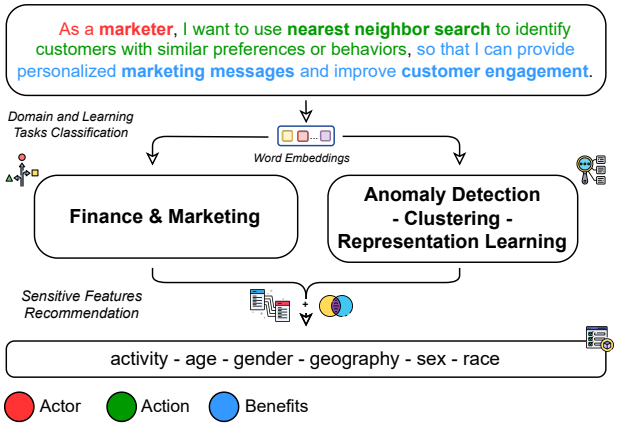


Figure 4.6: REFAIR: Running Example.

rectly classifies *Finance & Marketing* as domain and identifies *Anomaly Detection*, *Clustering*, and *Representation Learning* as possible ML tasks through which the US may be implemented. The classification of these tasks is consistent, as these are widely applied in the context of financial applications [64]. Finally, for each ML task, REFAIR maps relevant sensitive features, recommending *working activity*, *age*, *gender*, *geography*, *sex*, and *race* as the set of sensitive features to consider in the subsequent implementation steps of the US. The recommendations can be considered both complete and consistent, as these are the well-known attributes that should be considered while developing financial ML applications.

## 4.4 EMPIRICAL EVALUATION

The *goal* of our study was to assess REFAIR, with the *purpose* to measure the extent to which it may support fairness requirements engineering. The *perspective* is of researchers and practitioners. The former are interested in assessing the viability of automatically detecting sensitive features from User Stories. The latter are interested in assessing whether it may be actually employed in practice.

Specifically, we first focused on the capabilities of REFAIR in classifying application domains and machine learning tasks.

These are indeed the two aspects that determine the final accuracy of the recommendations provided, i.e., if ReFair correctly classifies application domains and machine learning tasks, the outcome will be correct *by definition*, as the sensitive features recommended would directly map onto the base ontology reporting the ground truth on the fairness attributes to consider in that domain and for those ML tasks. We formulated two research questions (**RQ**s):

🔍 **RQ$_{Fair}$1.** *To what extent can* ReFair *classify ML-specific application domains from User Stories?*

🔍 **RQ$_{Fair}$2.** *To what extent can* ReFair *classify ML-specific tasks from User Stories?*

After assessing the classification components of ReFair, we moved toward the evaluation of the sensitive feature recommender. When either the application domain or the ML tasks are misclassified, ReFair may recommend sensitive features that are inconsistent for a given US. Hence, we formulated a third research question:

🔍 **RQ$_{Fair}$3.** *To what extent can* ReFair *recommend sensitive features from User Stories?*

In terms of reporting we followed the guidelines by Wohlin et al. [232], other than the *ACM/SIGSOFT Empirical Standards*.[5] The *context* of the study was represented by the synthetic dataset described in Section 4.2. For each application domain, the dataset contained 365 USs which were used to experiment with our framework.

### 4.4.1 *Addressing **RQ$_{Fair}$1**: The ReFair Application Domain Classification Performance*

We addressed **RQ$_{Fair}$1** by experimenting with the word embeddings and domain classifiers supported by our framework.

**Experimental Setting.** We designed an empirical study to identify the best classifier among the 25 ML multiclass classification algorithms available within ReFair. Such an experiment was performed

5 Available at: https://github.com/acmsigsoft/EmpiricalStandards. Given the nature of the study and the standards currently available, we employed the *"General Standard"*, the *"Data Science"*, and the *"Engineering Research"* guidelines.

| Comparison of selected embedding techniques for the Domain Detection classifier. | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **TF-IDF** | | | **BERT** | | | **Word2Vec** | | | **FastText** | | | **Glove** | | | |
| Model | F1-Score | Accuracy | Model | F1-Score | Accuracy | Model | F1-Score | Accuracy | Model | F1-Score | Accuracy | Model | F1-Score | Accuracy | |
| ET | 0.80 | 0.80 | **XGBC** | **0.98** | **0.98** | CCCV | 0.91 | 0.91 | SVC | 0.94 | 0.94 | CCCV | 0.91 | 0.91 | |
| SVC | 0.80 | 0.80 | BC | 0.98 | 0.98 | LR | 0.91 | 0.91 | CCCV | 0.94 | 0.94 | LR | 0.91 | 0.91 | |
| CCCV | 0.80 | 0.80 | DT | 0.98 | 0.98 | LSVC | 0.90 | 0.90 | LR | 0.94 | 0.94 | LDA | 0.90 | 0.90 | |

Table 4.1: Domain classifier selection - Experimental Results. ET = Extra Trees, SVC = Support Vector Classification, CCCV = Calibrated Classifier CV, XGBC = XGB Classifier, BC = Bagging Classifier, DT = Decision Tree, LR = Logistic Regression, LSVC = Linear SVC, LDA = Linear Discriminant Analysis.

through the use of *Lazy Predict*,[6] a Python library that facilitates the comparison of multiple models and does not require manual parameter tuning. All the algorithms were evaluated using the five word-embedding techniques listed in Section 4.3.1. Specifically, the US dataset was first represented by using the $i^{th}$ embedding technique considered. Afterwards, we applied a ten-fold cross validation [72] to split the dataset in ten folds and let *Lazy Predict* assess the performance of each of the 25 algorithms on each fold. To effectively manage the computational demands associated with testing multiple combinations of word embeddings and ML techniques, we represented USs using a fixed vector of 100 tokens, independently from the size of the requirements or the embedding method used. This approach reduced the overall computational time required for testing while keeping the results informative and meaningful. The results were then analyzed using *pandas* [139], which allowed us to group them and obtain the average performance for all models over all folds. As the experiment focused on domain detection, which we modeled as a multiclass classification task, we used F1-Score and accuracy as evaluation criteria [174]. F1-Score is a commonly used metric for evaluating the performance of multiclass classification models [189]. It is computed as the harmonic mean of precision and recall, which provides a balance between these two measures, where precision measures the proportion of correct positive predictions among all positive predictions and recall measures the proportion

6 The *Lazy Predict* library: https://github.com/shankarpandala/lazypredict

of correct positive predictions among all actual positive instances. Accuracy measures the overall proportion of correct predictions, regardless of the class. It is computed as the ratio of correctly classified instances to the total number. At the end of this analysis, the best model was subject to a hyperparameters fine-tuning step to obtain the model that best fits the supplied data. Once we had identified the best performing classifier, we further refined it by running the RandomizedSearchCV algorithm: this is an automated configuration instrument provided by Scikit-learn, which involves testing random combinations of values from a range, as opposed to predefined values in classical Grid Search. This allowed us to (1) understand whether the results obtained through *LazyPredict* were reliable and (2) carry out another round of cross-validation, consolidating the results obtained and preventing them from being facilitated by the split used for the previous analysis.

**Experimental Results.** We evaluated 125 combinations of word embeddings and classification algorithms. While the detailed results for all models are available in our online appendix [184], Table 4.1 presents the top-3 models for each embedding technique in terms of accuracy and F1-Score. Notably, the XGB Classifier [41]—an implementation of gradient-enhanced decision trees designed for speed and performance—using BERT as word embedding emerged as the best-performing model, reaching 98% of F1-Score and accuracy. Two other models, i.e., Bagging Classifier and Decision Tree, also achieved the same performance, highlighting the effectiveness of BERT as word embedding for the textual representation of our task. Nonetheless, the combination of BERT and XGB Classifier provided the best compromise between performance and efficiency: hence, we deemed this model as the best one resulting from the application domain model selection step.

In the second step, we configured XGB Classifier by considering *max-depth*, *learning-rate*, *subsample*, and *n_estimators* as hyperparameters. The best configuration was the following: {*learning-rate: 0.087*, *max-depth: 3*, *n-estimators: 80*, *subsample: 0.924*}. The performance of the XGB Classifier were similar to those obtained without hyperparameter configuration. On the one hand, this confirmed that we could classify the application domain within USs with high accuracy.

On the other hand, our findings suggest that an additional, possibly costly fine-tuning refinement would not be strictly required to obtain high performance.

**Summary of the Results.** We empirically evaluated 125 ML algorithms and word embedding combinations to address $\mathbf{RQ}_{Fair}\mathbf{1}$. The combination of BERT and XGB CLASSIFIER exhibited Accuracy and F1-Score close to 98% in the domain detection task.

### 4.4.2 *Addressing $\mathbf{RQ}_{Fair}\mathbf{2}$: The ReFair Machine Learning Tasks Classification Performance*

Similarly to what done previously, we addressed $\mathbf{RQ}_{Fair}\mathbf{2}$ by experimenting with the word embeddings and machine learning tasks classification mechanisms supported by our framework.

**Experimental Setting.** As described in Section 4.3.2, the classification of the ML task associated to a fairness-critical application domain may be challenging, as multiple tasks can be used during the development of ML-enabled systems. As such, we modeled a multi-label classification problem, wherein USs may relate to multiple ML tasks. We leveraged SCIKIT-MULTILEARN [209], a BSD-licensed library for multi-label classification that is based on the well-known SCIKIT-LEARN ecosystem. We exploited *MultiLabelBinarizer* [209] to transform the output to be predicted. Unlike a single value indicating the class to which it pertains, in this case, the output is represented by an array of values indicating which classes the US belongs to. Similarly to $\mathbf{RQ}_{Fair}\mathbf{1}$, we experimented with each combination of multi-label technique, classification model, and word embedding method considered by REFAIR, by performing a ten-fold cross validation [72]. The results were examined using *pandas*, which allowed us to cluster the results and derive the mean performance for all possible combinations across all folds. As evaluation metrics, we considered F1-Score and Hamming Loss values for every combination. Both metrics have been widely adopted in the context of multi-label classification problems, especially when the number of labels is large [215]. In particular, the F1-Score was computed for each label separately, taking the average as a final performance indicator.

As for the Hamming Loss, this measures the fraction of misclassified labels. It was computed as the average number of labels that were incorrectly predicted for each instance.

Comparison of selected embedding techniques for ML Task Detection classifier.

| TF-IDF | | | BERT | | | Word2Vec | | | FastText | | | Glove | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Technique + Model | F1-Score | Hamming Loss | Technique + Model | F1-Score | Hamming Loss | Technique + Model | F1-Score | Hamming Loss | Technique + Model | F1-Score | Hamming Loss | Technique + Model | F1-Score | Hamming Loss |
| LP + RF | 0.88 | 0.06 | LP + DT | 0.86 | 0.07 | LP + LSVC | 0.86 | 0.07 | LP + LSVC | 0.81 | 0.09 | **LP + LSVC** | **0.90** | **0.05** |
| LP + DT | 0.82 | 0.09 | LP + RF | 0.84 | 0.08 | LP + GNB | 0.76 | 0.12 | LP + GNB | 0.81 | 0.10 | LP + GNB | 0.72 | 0.15 |
| BR + RF | 0.80 | 0.09 | BR + DT | 0.78 | 0.11 | BR + KNN | 0.68 | 0.15 | BR + KNN | 0.74 | 0.12 | BR + KNN | 0.67 | 0.15 |

Table 4.2: Machine Learning Task classifier selection - Experimental Results

**Experimental Results.** Overall, we empirically evaluated 90 combinations of word embeddings and ML task classification algorithms. While the complete results are in online appendix [184], Table 4.2 reports the top-3 performing models for each embedding technique. The combination of GLOVE, LABEL POWERSET, and LINEAR SUPPORT VECTOR CLASSIFICATION obtained the highest F1-Score and the lowest Hamming score. Nevertheless, unlike $\mathbf{RQ}_{Fair}\mathbf{1}$, where the word-embedding technique impacted the classification results, in this case, we discovered that the multilabel classification technique made the difference, achieving excellent results regardless of the word-embedding technique. More particularly, LABEL POWERSET, in combination with different classification algorithms, achieves results above 70% in terms of F1-Score with a Hamming Loss of at most 15% in the worst case, even with different techniques for text representation. The best results were finally obtained by combining LABEL POWERSET, GLOVE, and LSV: the F1-Score was slightly above 90%, while the Hamming Loss scored 5%.

🔑 **Summary of the Results.** We assessed 90 combinations of multilabel techniques, ML algorithms, and word-embedding methods. The combination of LABEL POWERSET, GLOVE, and LINEAR SUPPORT VECTOR CLASSIFICATION reached an F1-Score higher than 90% and Hamming Loss of 5% on the ML tasks classification.

### 4.4.3 *Addressing $\mathbf{RQ}_{Fair}\mathbf{3}$: The ReFair Sensitive Feature Recommendation Capabilities*

We addressed $\mathbf{RQ}_{Fair}\mathbf{3}$ by comparing the output of ReFair against an oracle reporting the actual set of sensitive features to be provided.

**Experimental Setting.** To address $\mathbf{RQ}_{Fair}\mathbf{3}$, we first built an oracle reporting the correct set of sensitive features for each US of our dataset. This was done by exploiting the base ontology [64]. As explained in Section 4.2.2, it provided sensitive features for each application domain and ML task: as the considered USs were concerned with an actual application domain and ML task, we could use the base ontology to label each US with the corresponding set of sensitive features. Afterward, we compared such an oracle against the recommendations of ReFair. This evaluation allowed us to measure how much the misclassifications assessed in $\mathbf{RQ}_{Fair}\mathbf{1}$ and $\mathbf{RQ}_{Fair}\mathbf{2}$ altogether influenced the capabilities of our framework, hence providing a final assessment of the support that ReFair may provide during requirements engineering. We computed the MoJo distance [216] as the evaluation metric, which is a widely accepted method to measure the distance between two partitions of the same set. This metric ranges from 0 (indicating a completely identical set) to 1 (representing two completely different sets). It is based on the number of "moves" required to make the two compared sets identical. Here, a "move" refers to either shifting a single data point from one cluster to another or swapping the cluster assignments of two data points. To evaluate the distance between the set of features recommended by ReFair and the oracle for each US, we employed a specific MoJo variant that relies on the Jaccard's and the Mean sets overlap indexes [143]. In addition, we analyzed the results from a more qualitative standpoint by computing the *amount* of sensitive features erroneously recommended by ReFair: to this aim, we leveraged the token-level Levenshtein distance [145], considering each token as a sensitive feature.

**Experimental Results.** The average MoJo distance computed by comparing all the sets of sensitive features against the oracle reached 0.04, meaning that the output of ReFair was just 4% far from the ideal one. This result indicates that the misclassifications in terms

of application domain and ML task have a marginal impact on the overall capabilities of ReFair, hence making it a potentially suitable instrument to support requirements engineering activities. Going deeper, the Levenshtein distance [145] analysis revealed that ReFair acted as a *perfect recommender* on 11,969 USs (97%). Of the remaining 432 (3%), in 41 cases (less than 1%) the feature sets of ReFair and the oracle differed by only 1 feature, 70 (0.6%) differed by 2 features, and 321 (less than 3%) differed by more than 2 features. By analyzing the latter cases more closely, we observed that (1) these would have still led ReFair to provide some support in a real case scenario, as the set of recommendations were partially correct, i.e., wrong recommendations were up to 52.5% of the recommendations provided by ReFair; and (2) they were mostly due to specific application domains, e.g., *Health*, and ML tasks, e.g., *Classification* which were those more often misclassified. This suggests that further improvements of ReFair could revolve around the addition of targeted data samples or the application of data augmentation methods able to provide the framework with a more consistent knowledge. We report additional analyses into the capabilities of ReFair per application domain and ML task as part of the technical report that accompanies this submission [184].

🔑 **Summary of the Results.** The MoJo distance showed a near-perfect match (0.04) between the set of sensitive features recommended by ReFair and by the oracle. Our framework acts as a perfect recommender in 90% of the cases, providing mostly wrong recommendations in just 5% of the cases.

## 4.5 THREATS TO VALIDITY

Our study suggested that ReFair may represent a valuable instrument, yet various aspects might have biased the conclusions drawn.

**Threats to External Validity.** A first threat is represented by the user story format employed in the study. While this possibly limits the applicability of ReFair, previous studies showed that these are widely used in the requirements engineering of machine learning-

enabled systems [223]. As part of our future research agenda, we plan to generalize the framework to other formats.

Our framework was built on top of the current knowledge on the fairness-critical application domains and machine learning tasks [60, 64]. As such, its application is limited to such a knowledge. However, ReFair was designed to be easily extended; the source code is publicly available so that researchers can build on top of it.

The generalizability of the results might have been threatened by the granularity of the USs automatically generated and, more in general, by the use of synthetic USs to experiment with ReFair. As explained in Section 4.2.3, we accounted for the liberty developers have in real-case scenarios [122], hence enabling the generation of both solution- and problem-oriented USs that might have simulated a realistic use case for ReFair. Nonetheless, further investigations into the generalizability of these USs should be pursued. To partially mitigate such a threat to validity and preliminarily assess how ReFair may work in a realistic environment, we conducted a *qualitative* experimentation involving real-world ML engineers. The goal was to understand the capabilities of our framework when run against manually-written requirements specifications. We involved 20 ML engineers from our contact network and asked them to develop requirements specifications that could be later employed to assess the soundness of our framework. The participants had between two and five years of professional experience, had knowledge on both software engineering and artificial intelligence. We involved them through e-mails, by asking for a volunteer participation.

Upon confirmation of their participation, we sent them a link to an online questionnaire which comprised three sections. The first presented the informative consent: we clarified that the answers would have kept anonymous to preserve privacy and that their responses would have been used for a research submission. The second aimed at collecting demographic data. The third proposed a problem statement concerned with a specific machine learning domain among those investigated in this study [64]. For instance, one of the problem statements revolved around the definition of a ML-based software system able to classify cancer types based on genomic data. While each participant was assigned to a problem

statement pertaining to a different domain, we could not assess all 34 domains considered in the study because of the lack of participants.

The problem statements were crafted by the first author, who elaborated them with the help of online resources and existing projects, in an effort of producing realistic cases to propose to the participants. The full set of problem statements are available in our online appendix [184].

Participants were asked to carefully read the problem statement they were assigned to and produce up to ten requirements involving the machine learning solutions that may be employed in the context. We gave them ten days to deliver, collecting a set of 119 requirements that could be classified by ReFair, which was configured according to the empirical results presented in Section 4.4.

We could not compute precise performance metrics because of the unavailability of a ground truth for these requirements, yet we manually went through a qualitative analysis and assessed whether these might have been considered similar or meaningful. We noticed that the major differences between the manually- and automatically-generated USs were due to two aspects. On the one hand, the participants' knowledge on the problem statements presented: when participants were not familiar enough with the domain, they indeed developed more generic USs that were hard for ReFair to correctly handle. On the other hand, the machine learning engineers' experience with requirements engineering: we noticed that more experienced practitioners were able to develop higher-quality USs that better characterized the elements to be implemented, hence allowing ReFair to properly classify potential sensitive features. In conclusion, the qualitative investigation confirmed the value of ReFair, yet discovered the potential boundaries that may affect its capabilities, namely *domain familiarity* and *experience* of the practitioners that engage with the framework. As part of our future research agenda, we aim at performing larger-scale studies that might better assess the differences between manually- and automatically-written USs, other than the implications for the classification performance of ReFair.

**Threats to Construct Validity.** Our study was designed to take the context-dependent nature of software fairness into account, i.e., some sensitive features might be considered as such depending on

the context: as such, we trained and tested ReFair on USs coming from a large variety of fairness-critical domains [64].

Additional, hardly operationalizable external factors, e.g., laws and regulations, may influence the identification of sensitive features: as any recommendation system, ReFair must be considered as an assistant rather than a tool to replace the requirements engineer by providing insights that might be relevant for the development of USs.

As for the synthetic dataset, we built it on top of the existing knowledge on requirements engineering [205, 223] and software fairness [64], favoring the generation of USs having different levels of granularity [122] and employing a reliable Large Language Model like ChatGPT, which we inquired only after experimenting with multiple prompts. The internal validation of the dataset quality, other than the external validation conducted with practitioners, increased our confidence on its validity and suitability for our purposes. Replications of our study on real-world datasets would still be desirable and part of our future research agenda.

We built a prototypical implementation of ReFair by experimenting with a wide set of shallow machine learning algorithms to classify application domains and tasks within user stories. We did not make use of advanced artificial intelligence solutions, e. g.deep learning algorithms. While they might have offered additional insights into the capabilities of our framework, we favored the analysis of simpler models which are less demanding in terms of computation costs and training data, other than being more interpretable. The high classification performance obtained by those simpler models increased the confidence of our design choices, even though we plan to investigate the contribution of more complex classification models as part of our future research agenda.

**Threats to Conclusion Validity.** We assessed the classification components of ReFair by experimenting with multiple classifiers and word embedding techniques, computing well-established metrics, i.e., F1-Score and accuracy, that have been widely used to comprehensively assess multiclass and multilabel classification algorithms [189, 215]. As for the sensitive feature recommender, we defined evaluation metrics that could well represent the capabilities of ReFair in recommending appropriate sensitive features.

Another threat concerns the use of the same dataset to train and test our approach. In this respect, there are three considerations to make. First, we had no alternatives than using the synthetic dataset, given the lack of alternatives in literature. Second, we made sure not to mix training and testing data by performing a ten-fold cross validation: in each iteration one fold was retained as the test set and left untouched, while the remaining folds were used for training. By doing that, we ensured the ReFair was always experimented against unseen data. Last but not least, after addressing our research questions relying on the synthetic dataset, we performed a more qualitative investigation into the performance of ReFair on a manually-generated dataset of USs. This qualitative analysis was explicitly designed to verify how ReFair may work when applied on a dataset different than the one used for training. The conclusions drawn in our qualitative study still pointed out the promising performance of our approach: of course, we are aware of the need for further, larger-scale evaluations of ReFair but, at the same time, we believe that the results provided so far represent a valuable contribution to the research community.

# 5 BEYOND DOMAIN DEPENDENCY IN SECURITY REQUIREMENTS IDENTIFICATION

## 5.1 INTRODUCTION

Effective requirements gathering is essential for successful software development, as inaccuracies or omissions can disrupt the entire software lifecycle [172, 203]. While stakeholder communication mitigates some challenges in requirements engineering (RE), it is often insufficient for identifying non-functional requirements (NFRs), particularly security requirements. These requirements are challenging to discern due to stakeholders' limited expertise and their implicit presence in documentation [110, 159, 166, 198]. The importance of security as an NFR is underscored by the substantial financial impact of security breaches [8], necessitating explicit identification and specification of security requirements, which are often implicitly distributed across requirement specifications [104, 117].

The manual identification of security requirements is error-prone and time-consuming, highlighting the need for automation. Current research predominantly employs supervised machine learning (ML) methods [48, 104, 117], which prioritize statistical lexical features over syntactic structures and semantic relationships. Consequently, these models often struggle to distinguish nuanced contextual meanings, leading to limited generalizability across diverse domains due to their reliance on large, domain-specific datasets [151]. To address domain-specific limitations, Munaiah, Meneely, and Murukannaiah [154] proposed a one-class classification model trained on the Common Weakness Enumeration (CWE) database. While this approach demonstrated the potential of domain-independent approach, it relied solely on CWE descriptions, leaving questions about the role on the impact of using broader vulnerability datasets unanswered. Furthermore, its reliance on One-Class SVM (Support Vector Machine)

highlights the need for exploring alternative classification techniques to improve and facilitate the security requirements identification task.

Recognizing these previous findings, in this work we assess the effectiveness of shallow and advanced techniques for detecting security requirements, emphasizing ML and BERT-based models that offer the capability of fine-tuning for various NLP-based tasks, including sequence classification [57, 221].

We first assess if shallow ML models can identify security requirements within the same domain, motivating our first RQ:

**RQ$_{Sec}$1. How effectively can shallow machine learning algorithms, based on word embeddings, identify security-related requirements coming from the same domain?**

A positive answer to RQ$_{Sec}$1 provides an alternative approach with respect to the methodologies previously implemented in [48, 104, 117], and avoiding the dependency on extensive datasets for model training as done in [48, 151]. To address RQ$_{Sec}$1, we extensively experiment with 5 embedding techniques and 30 ML algorithms, totaling 150 solutions. After identifying the most effective combinations, we optimize hyperparameters and leverage ensemble learning (stacking, bagging, voting) in attempt to improve predictive performance. We evaluate the accuracy on datasets of requirements previously used in literature, i.e., Common Electronic Purse (ePurse), Customer Premises Network (CPN), and the Global Platform Specifications (GPS) [104, 117]. In the following, we will refer to this setup experiment with the term *intra-domain*, as we train and validate the models on requirements coming from the same domains.

We assess the generalizability of the approach by testing the models on datasets different from those used for training (meaning that requirements come from another domain). To address this aspect of the research, we formulate the following research question:

**RQ$_{Sec}$2. How effectively can shallow machine learning algorithms, based on word embeddings, identify security-related requirements coming from different domains?**

It should be noted that our methodology intentionally limits the number of projects used to better replicate real-world conditions, where typically few projects are available. This contrasts with the approach taken in [151], which utilized 15 diverse projects, of which

only three were from the industrial sector. We consider the work of Li and Chen [117] as the baseline for these comparisons, as it represents the best result achieved in this scenario.

To evaluate $RQ_{Sec}2$, we train several WE and ML combinations on 6 datasets formed by 3 individual collections of requirements (CPN, ePurse, and GPS), as well as the 3 combinations formed by combining two of the three collections (CPN & ePurse, CPN & GPS, ePurse & GPS). The trained models are used to classify security requirements of datasets not included for training, performing hold-out validation. $RQ_{Sec}2$ exploration finds optimal hyperparameters for the top combination, and tests ensemble learning strategies as in $RQ_{Sec}1$. In the following, we will refer to this experiment with the term *inter-domain*, as we train the models on requirements coming from a particular set of domains and validate on requirements from different domains.

The challenge of acquiring datasets that are independent of specific domains becomes evident, as they are scarce and demand extensive curation by domain experts. To overcome this constraint, we broaden our search beyond the problem domain to include solution domains, specifically focusing on the CWE and Common Vulnerabilities and Exposures (CVE). CWE serves as a formal catalog of software weakness types, while CVE provides a standardized list of known cybersecurity vulnerabilities. This method mirrors that of [154], where a one-class classification model was trained on descriptions from the CWE to contextualize system weaknesses. Building on this concept, we formulate the following research question:

**$RQ_{Sec}3$. How effective can pre-training context in BERT transformers be on the detection of security-related requirements?**

To address $RQ_{Sec}3$ we pre-trained three distinct BERT models, each leveraging different datasets: (1) descriptions from CWE and the CVE; (2) the Defect Detection dataset [239] tailored for the Insecure Code Detection task; and (3) a combination of both datasets. This strategy, particularly the use of BERT's pre-training phase, addresses the challenge of detecting inter-domain security requirements in scenarios where diverse project data for training is scarce. In fact, BERT's pretrained representations capture rich semantic information from large text corpora, allowing the model to generalize well to

diverse tasks with minimal finetuning. This bridges the gap between requirement specifications and vulnerability descriptions, paving the way for a robust inter-domain classifier to advance automated security requirement identification.

In summary, the contributions of this work are:

1. Benchmarking shallow and deep learning approaches including ML and fine-tuned BERT for security requirement detection.
2. Investigating combined word embeddings and ML algorithms for intra-domain security requirement identification.
3. Testing real-world applicability by evaluating inter-domain security requirement classification.
4. Recognizing labeled cross-domain data scarcity, pre-training BERT on security databases for contextual knowledge to identify requirements expressed differently across projects.
5. A publicly accessible replication package [6] is provided, enabling researchers to reproduce our findings or to further develop upon our work.

## 5.2 EMPIRICAL STUDY DESIGN

In this section, we outline the empirical study design, starting with the rationale for formulating the research questions. We then detail the datasets used for experimentation, the evaluation criteria for predicting accuracy of the considered approaches, and discuss potential threats to the study's validity.

### 5.2.1 *Research questions*

This study aims to assess the effectiveness of shallow and advanced techniques for detecting security requirements, particularly by exploiting ML algorithms and BERT-based models, which allow finetuning for different NLP-based tasks, such as sequence classification.

We start by assessing whether shallow ML models can be used for the task, which led to the formulation of the first research question:

**RQ$_{Sec}$1** How effectively can shallow machine learning algorithms, based on word embeddings, identify security-related requirements coming from the same domain?

The formulation of this research question aims to understand which of the possible combinations of WE and ML algorithms presents the best performances by following the strategy shown in Figure 5.1.

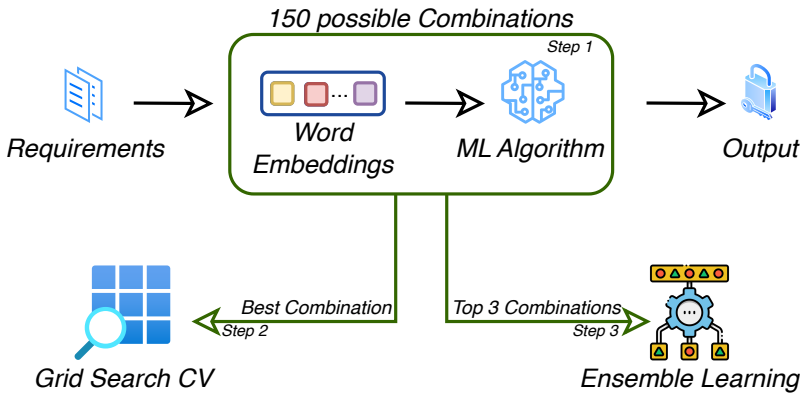


Figure 5.1: Procedure employed to address RQ$_{Sec}$1 and RQ$_{Sec}$2.

We experiment with 5 embedding techniques and 30 ML algorithms, totaling 150 possible solutions, to find the best combination. The word-embedding techniques we adopt are TF-IDF, Word2Vec, GloVe, FastText, and BERT. The choice and experimentation of these techniques are not random, as each technique has its peculiarities and other studies have used the same ones (e.g., [67]).

TF-IDF [182] is based on statistics of the frequency of words in a document and their importance concerning the entire corpus. It is characterized by its simplicity and interpretability, being effective for problems in extracting information from textual documents. Word2Vec [146, 148] is known for capturing semantic relationships between words via dense word vectors. It is particularly suitable for applications that require a deeper understanding of word meaning,

such as semantic clustering and similarity analysis. GloVe [169] combines co-occurrence statistics of words in a corpus with matrix-based optimization. It effectively captures global relationships between words and retains information on the semantic structure of the data. FastText [20, 92, 93] is known for its ability to handle subword information, making it suitable for languages with rich morphology. It is ideal for classifying texts and searching for similar documents with common roots. BERT [57] is a transformer-based embedding model that can capture the bidirectional context of words. It is particularly effective in tasks such as machine translation and text summarization.

We empirically investigate the optimal classifier from the 30 different classification algorithms provided by *Lazy Predict*[1], a Python library streamlining the comparison of numerous models, obviating the need for manual parameter tuning. The assessment of all these algorithms is conducted employing the five aforementioned word-embedding techniques. After identifying the most effective combinations (Step 1 in Figure 5.1), our analyses encompass two aspects: the first facet of our investigation is to optimizing hyperparameters for the selected ML model (Step 2 in Figure 5.1), and the second revolves around implementing ensemble learning techniques leveraging the top three combinations (Step 3 in Figure 5.1). The pursuit of optimal hyperparameters is realized through Grid Search Cross-Validation (GridSearchCV) [168], a well-established and rigorous method renowned for its exhaustive exploration of the hyperparameter space. This process is designed to fine-tune the selected ML model, enhancing its precision and adaptability to the specific task at hand. In the realm of ensemble learning, we explore stacking [233], bagging [23], and voting methodologies [241]. It is worth noting that in this analysis, we employ data coming from a collection of industry requirements of three different projects: ePurse, CPN, GP. Details about the datasets can be found in Section 5.2.2. These datasets have been chosen because previous studies exploited them [104, 117], allowing us to carry out a comparison with existing classification models.

1 The *Lazy Predict* library: https://github.com/shankarpandala/lazypredict

In particular, as in previous investigations, we perform two types of analysis:

(a) when the models are trained and validated on the same set of requirements (i.e., intra-domain) and

(b) when the models are trained on a collection of requirements and then validated on a different set of requirements (i.e., inter-domain).

Thus, with $RQ_{Sec}1$, we want to understand which of the analyzed approaches presents the best performance when evaluated on data coming from the same domain of the training dataset, addressing point (a). Having three different sets of requirements from Knauss et al. [104] and Li and Chen [117], we validate the built models on each of these datasets and their combinations for a total of seven datasets. The seven combinations are obtained using the three individual datasets (CPN, ePurse, and GPS), the three combinations obtained using each time two of the three collections of requirements (CPN & ePurse, CPN & GPS, and ePurse & GPS), and the combination obtained using all of them (CPN & ePurse & GPS). We apply a *10*-fold cross-validation on each of the datasets and their combinations, using at each iteration a different set for training and testing starting from the whole set of requirement considered, to verify that the obtained results are not the product of chance (see Section 5.2.2).

To address point (b), we formulate the following research question:

> **$RQ_{Sec}2$** How effectively can shallow machine learning algorithms, based on word embeddings, identify security-related requirements coming from different domains?

To answer $RQ_{Sec}2$, the WE and ML algorithm combinations are trained on six datasets. These datasets are formed by combining three sets of requirements collections (CPN, ePurse, and GPS) and three combinations derived from pairing two of the three requirement collections each time (CPN & ePurse, CPN & GPS, and ePurse & GPS). The obtained models are then used to identify the security requirements of the collection not included in the training set (e.g., the models trained on ePurse are used to classify the requirements

included in CPN and GPS, the models trained on CPN & GPS are used to classify the requirements included in ePurse), thus performing a hold-out validation. See Section 5.2.2 for details regarding the complete list of training and test sets employed.

As in the case of $RQ_{Sec}1$, we conduct a deeper analysis to identify the optimal hyperparameters for the leading combination and practical experiments involving various ensemble learning strategies.

Thus, with $RQ_{Sec}2$, we want to evaluate the generalizability of the approaches considered in $RQ_{Sec}1$. For this reason, the training and test data must belong to different domains.

However, obtaining training datasets for multipe domains is challenging. Creating such datasets involves the laborious task of having domain experts classify requirements into security and non-security categories, which is a time-consuming process. Consequently, reliance on datasets coming from a particular domain can restrict the practical applicability in a real-world scenario for automated security requirement identification.

To address this limitation, we broadened our focus beyond the problem domain to include solution domains, leading us to consider the Common Weakness Enumeration (CWE) and Common Vulnerabilities and Exposures (CVE) as potential training datasets[2]. CWE provides a structured catalog of software weakness types, offering a standardized language for describing security vulnerabilities in architecture, design, and code. Similarly, CVE is a comprehensive list of documented cybersecurity vulnerabilities, widely used for referencing and tracking issues across software and hardware systems.

It is worth noting that software weaknesses can be seen as the aftermath of unmet security requirements. We consider the language employed to characterize security requirements and the one used to delineate weaknesses share commonalities, making it conceivable to train a inter-domain security requirements classifier using the descriptions of weaknesses and vulnerabilities. Take, for example, a requirement in the GPS dataset, such as "*When relevant, a Security Domain must verify the signature of load file data blocks upon request from the OPEN*". This requirement displays a resemblance to the descrip-

2 https://cwe.mitre.org/index.html

tion of CWE-347, which articulates that "*The product does not verify, or incorrectly verifies, the cryptographic signature for data.*" (example used also by Munaiah, Meneely, and Murukannaiah [154]). The convergence in both language and content between these two instances strengthens our hypothesis that an approach capable of identifying security requirements in a domain agnostic manner is indeed viable. To this end, the following research question is formulated:

**RQ**$_{Sec}$**3** How effective can pre-training context in BERT transformers be on the detection of security-related requirements?

We leverage an extensive corpus of textual descriptions as well as insecure code related to software weaknesses and vulnerabilities to investigate the feasibility of training a domain-independent security requirements classifier. We recognize that the descriptions and the code in CWE and CVE databases contained rich and comprehensive information about various security-related issues in software.

We initiate the pre-training of a BERT model to harness this valuable textual data. BERT is an advanced natural language processing model renowned for its ability to grasp contextual information from extensive text datasets, enabling effective generalization to various tasks with minimal fine-tuning. In this case, we pre-train the BERT model using the textual descriptions from CWE and CVE, and the code from CodeXGLUE Defect Detection dataset [239], allowing it to understand the nuanced language used to describe software weaknesses and vulnerabilities as well the related portions of code. See Section 5.2.2 for details regarding the descriptions of weaknesses and vulnerabilities. This pre-training step is crucial in preparing the BERT model to identify security requirements across diverse domains effectively. The model is exposed to various security-related codes, languages and terminology, enabling it to recognize the patterns between security requirements and software vulnerabilities.

By employing this approach, as shown in Figure 5.2, we aim to bridge the gap between the language used to express security requirements and the descriptions of known vulnerabilities, ultimately paving the way for a more robust and domain-independent security requirements classifier.

For this reason, we end with four distinct BERT models:

- **BERT** ((1) in Figure 5.2): The specific text corpus used for pre-training BERT can vary depending on the variant of BERT and the objectives of the pre-trained model. The original BERT model, developed by Google AI, was pre-trained on a combination of the BooksCorpus dataset (consisting of 11,038 books) and the English Wikipedia (comprising approximately 2,500 million words). In this study, we utilize the BERT base uncased version, typically pre-trained on a substantial volume of text data collected from diverse sources on the web, including books, articles, websites, and more.

- **CweCveBERT** ((2) in Figure 5.2): Derived from the BERT base uncased model, we augment the pre-training phase by incorporating descriptions of CWE and CVE.

- **CodeBERT** ((3) in Figure 5.2): CodeBERT [65] is a bimodal pre-trained model designed to understand and bridge natural language (NL) and various programming languages (PL), such as Python, Java, JavaScript, and others. CodeBERT captures the semantic relationships between natural language and programming language and generates versatile representations suitable for a wide range of NL-PL understanding tasks (e.g., natural language code search) and generation tasks (e.g., code documentation generation). CodeBERT was pre-trained using code repositories from GitHub in six different programming languages. The bimodal data points in this model consist of pairs of source code and their corresponding function-level natural language documentation. In this work, we employ a version of CodeBERT fine-tuned on the CodeXGLUE Defect Detection dataset [239] for the Insecure Code Detection downstream task.

- **CweCveCodeBERT** ((4) in Figure 5.2): This model is a combination of the two previously mentioned BERT models, i.e., CweCveBERT and CodeBERT. Specifically, we initiate from CodeBERT and extend the pre-training process to include a text corpus composed of descriptions from both CWE and CVE.

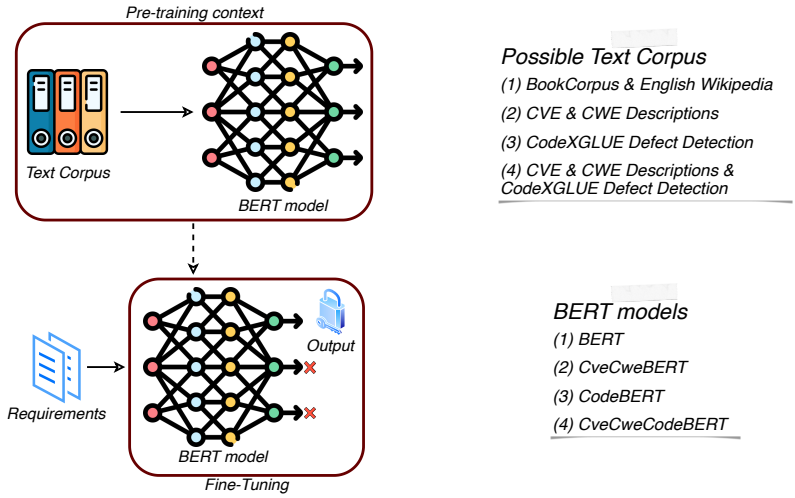


Figure 5.2: Procedure employed to address $RQ_{Sec}3$.

### 5.2.2 *Datasets*

Table 5.1 provides an overview of the sets of requirements considered. The datasets comprises lists of requirements specified for their respective projects. The sole exception is the United Nations regulations dataset ($Test_{Data}$), which combines three documents from the automotive sector (UN-R155, UN-R156, and UN-R157). These regulations differ in their level of security relevance. Most of the individual requirements consist of only one sentence, while others contain two sentences, with an average number of words per requirement indicated in the last column in Table 5.1. Below is an example extracted from $Promise_{exp}$:

"*The product shall ensure that it can only be accessed by authorized users. The product will be able to distinguish between authorized and unauthorized users in all access attempts*".

The advantages of using these sets of requirements are:

- different approaches have been evaluated on these sets (e.g., Knauss et al. [104]; Munaiah, Meneely, and Murukannaiah [154]; Li and Chen [117]; Mohamad et al. [151]) allowing us to perform a direct comparison with our proposal;

Table 5.1: Requirements specifications considered in our empirical study.

| Dataset (Abbreviation) | Total Requirements | Security Requirements | Avg. Words per Requirement |
|---|---|---|---|
| Common Electronic Purse (ePurse) [104, 117, 151] | 124 | 83 | 29 |
| Customer Premises Network (CPN) [104, 117, 151] | 141 | 31 | 24 |
| Global Platform Specifications (GPS) [104, 117, 151] | 174 | 63 | 31 |
| Collection of 12 projects ($Train_{Data}$) [151] | 3369 | 652 | 25 |
| United Nations regulations ($Test_{Data}$) [151] | 362 | 47 | 43 |
| PROMISE expanded version ($Promise_{exp}$) [120] | 969 | 125 | 18 |

- the specifications cover different application domains, allowing to evaluate the generalizability of the trained models;
- the requirements specifications in CPN, ePurse, and GPS come from industrial environments, and the success of their classification can demonstrate the usefulness of our approach in industrial settings.

To address the challenges in validating machine learning models, a dataset must be carefully managed, ensuring (*i*) the use of distinct data for training and validation, (*ii*) systematic selection of training requirements for reproducible and representative results, and (*iii*) avoidance of overfitting to prevent models from being limited to specific training data. Typically, *k*-fold cross-validation is employed to mitigate these issues, randomly sorting the dataset into *k* equal parts, training the classifier on *k - 1* parts, and evaluating its performance on the remaining part [38, 86, 227].

The choice of *k* is critical: a low value for *k* makes the evaluation less complicated and faster, characterized by a low variance but with the possibility of having more biases. Therefore, we should determine *k* experimentally to ensure unbiased results, while minimizing the complexity of our evaluation. According to Chantree [38], Knauss et al. [104], and Li and Chen [117], *k*=10 is an appropriate value for this purpose. With a value of *k* larger, the partitions may become too small with the risk of not containing a security-relevant requirement.

Thus, in our study, we apply a *k*(=10)-fold cross-validation to assess the performance of the proposed solutions. It is worth noting that the *k*-fold cross validation has not been applied in the experiment

for the $RQ_{Sec}2$ and to assess the BERT-based models built to answer $RQ_{Sec}3$ because they require that the test and training data belong strictly to different domains.

To systematically explore the hyperparameter space, GridSearchCV, a technique for hyperparameter tuning in ML, is employed. GridSearchCV automates the process of systematically testing different combinations of hyperparameters to find the best set of hyperparameters for a given model. This approach ensures that the model's performance is optimized with respect to the selected hyperparameters. It combines grid search, which exhaustively explores a predefined set of hyperparameter combinations, with cross-validation, which assesses model performance to avoid overfitting.

Grid search with cross-validation combines these two techniques by performing grid search on each fold of the cross-validation. For each hyperparameters combination, the model is trained $k$ times, and the performance metrics are averaged over the $k$ iterations. The outcome of grid search with cross-validation is the combination of hyperparameters that yields the best performance on the validation data. This approach helps in optimizing a model's hyperparameters and ensures that the selected configuration is less likely to be overfit to a specific dataset.

Taking into account these validation methods, to answer $RQ_{Sec}1$ and $RQ_{Sec}2$, we consider the first 3 sets, i.e., CPN, ePurse and GPS, and their combinations to train and test the different combinations of word embedding and ML algorithm (see Table 5.2). The remaining datasets, i.e., $Train_{Data}$, $Test_{Data}$, and $Promise_{exp}$, are useful for evaluating the BERT-based approaches built to answer $RQ_{Sec}3$. $Train_{Data}$ have been used to finetune the different BERT models and $Test_{Data}$ and $Promise_{exp}$ to test them.

Before evaluating BERT models we pre-trained them on the CWE[3] and CVE[4] descriptions catalogs. Below, we provide two examples of descriptions that help identify the displayed requirement from $Promise_{exp}$ as security-related:

3 We downloaded it at: https://cwe.mitre.org/data/downloads.html
4 We downloaded it at: https://www.cve.org/Downloads

**CWE-ID 203**:"*The product behaves differently or sends different responses under different circumstances in a way that is observable to an unauthorized actor, which exposes security-relevant information about the state of the product, such as whether a particular operation was successful or not.*".

**CVE-2020-0015**:"*In onCreate of CertInstaller.java, there is a possible way to overlay the Certificate Installation dialog by a malicious application. This could lead to local escalation of privilege with no additional execution privileges needed. User interaction is needed for exploitation.*".

Table 5.2: List of datasets used for training and test in each research question.

| **RQ$_{Sec}$1** | **RQ$_{Sec}$2** | **RQ$_{Sec}$3** |
|---|---|---|
| **10-fold cross-validation** | **Training set (Test set)** | **Different Pre-training setup (Figure 5.2)** |
| CPN | CPN (GPS) | **Fine-tuned on Train$_{Data}$ & CPN & GPS & ePurse** |
| GPS | CPN (ePurse) | Test$_{Data}$ |
| ePurse | ePurse (CPN) | Promise$_{exp}$ |
| CPN & GPS | ePurse (GPS) | |
| GPS & ePurse | GPS (ePurse) | **Fine-tuned on Train$_{Data}$** |
| CPN & ePurse | GPS (CPN) | CPN |
| CPN & GPS & ePurse | CPN & ePurse (GPS) | GPS |
| | CPN & GPS (ePurse) | ePurse |
| | GPS & ePurse (CPN) | |

### 5.2.3 *Evaluation criteria*

To assess the accuracy of our solutions, we employ widely used information retrieval criteria: Accuracy, Precision, Recall, and F1-score [9, 173].

***Accuracy*** It measures the proportion of total correct predictions (both *true positives* and *true negatives)* out of all predictions made by the model (*true positive + true negative + false positive + false negative*).

***Precision*** Calculated as *true positive* / (*true positive* + *false positive*), it indicates the correctness of the predictions provided by the model.

***Recall*** Calculated as *true positive* / (*true positive* + *false negative*), it measures the completeness of the predictions provided by the model.

***F1-score*** Defined as the harmonic mean of Precision and Recall, it indicates the balance between these two measures. We consider the weighted-averaged F1-score, calculated by taking the mean of all per-class F1-scores while considering each class's support.

We use these criteria because they are considered equally important in a binary classification task. The datasets used in this study enable the quantification of these criteria, as they include information about the actual class labels. For our study, the higher Recall could be considered more important than higher Precision, because stakeholders may be more interested in a tool that can capture all possible security requirements rather than focusing on the correctness of predictions [104]. In addition, these criteria allow comparison with previous work in the literature, leading us to understand and analyze the effectiveness and importance of our approach. The goal is to try to observe how accurate the models are in identifying security-related requirements and to catch their limitations in the above task.

As for the evaluation of the achieved results, to draw conclusions we consider a model characterized by an F1-score value greater than 0.7 to be good, as other work has also used this threshold as an index of model goodness [86, 104].

Furthermore, to establish if one model allows obtaining significantly better predictions than other models, we perform statistical tests aiming at assessing whether the differences observed by applying the chosen evaluation criteria (i.e., Accuracy, Precision, Recall, and F1-score) are legitimate or due to chance [226]. Specifically, we use McNemar's test to compare the performances of our prediction models [89, 192]. Indeed, Japkowicz and Shah [89] and Salzberg [192] recommend McNemar's test to compare the performances of two models because it has a lower probability of error because it

makes fewer assumptions. In particular, given the predictions of two models, $A$ and $B$, and truth labels, a contingency table is calculated to examine the number of cases in which the following occurs: (*i*) Both classifiers are correct; (*ii*) Both classifiers are incorrect; (*iii*) $A$ is correct and $B$ is incorrect; (*iv*) $B$ is correct and $A$ is incorrect. Through this table, it is possible to estimate the probability that $A$ is better than $B$ at least as many times as observed in the experiment [192].

To compare the performance of our classifiers, the following null hypothesis is considered:

*$Hn_0$: The considered BERT-based models are equally accurate in identifying security requirements*.

McNemar's test allows us to test the null hypothesis by comparing each pair of models under the same null hypothesis. As usual, for the performed statistical tests we consider accepting a probability of 5% of committing a Type-I-Error [232], thus a p-value of 0.05 as a threshold of "significance", i.e., a p-value less than 0.05 implies that the results obtained are unlikely to be due to chance, allowing the null hypothesis to be rejected. It is important to note that in our study, to answer $RQ_{Sec}3$ we compare three models [5], thus to verify if one model can provide significantly better predictions than the other two models, we apply two tests. As a consequence, we apply a Bonferroni correction to classify the p-values as significant (i.e., to reject the null hypothesis and positively answer our research questions) [52, 58]. Since the number of tests is two, the correction applied is: $0.05/2 = 0.025$ in the case of $RQ_{Sec}3$.

## 5.3 RESULTS AND DISCUSSION

In this section, we present the outcomes of our investigation and provide the answer to each research question posed.

5 Note that we implemented four BERT models but we "discarded" CweCveBERT, i.e., the one pre-trained on solely CWE and CVE descriptions catalogs, because in the first round of evaluation is the one that performed the worst, thus we excluded it in finding the statistical significant difference.

### 5.3.1 $RQ_{Sec}1$ *How effectively can shallow machine learning algorithms, based on word embeddings, identify security-related requirements coming from the same domain?*

Table 5.3 shows the overall results (in terms of Accuracy and F1-score, indicated as Acc and F1 in the table, respectively) obtained by the built combinations of WE and ML models on the different datasets, i.e., all the possible combinations of the three requirement specifications considered (CPN, GPS, ePurse, CPN & GPS, GPS & ePurse, CPN & ePurse, CPN & GPS & ePurse), to answer $RQ_{Sec}1$. We apply 10-fold cross-validation, and then we calculate the average Accuracy and F1-score to catch the most promising combination. Note that we consider only these 2 metrics because in the first part of the assessment we are interested in the general performance of the models, that can be resumed in Accuracy and F1-Score, while specific metrics like Recall or Precision will be considered in the later steps. Based on our result, the best combination is Word2Vec as word embedding technique and NuSVC as ML algorithm, reaching 0.84 in terms of Accuracy and F1-Score. Since the first part of the experiment only gives an overview of the possible best combination, we carried out more in-depth experiments both via *GridSearchCV*, to identify the best hyperparameters for the best combination identified, i.e. Word2Vec and NuSVC, and via Ensemble Learning techniques, with the aim of filling in the model's deficiencies on certain datasets by exploiting the learning of other algorithms. Table 5.4 presents the results obtained through *GridSearchCV*.

In this experiment, the *nu* parameter is tested at five levels: 0.1, 0.3, 0.5, 0.7, and 0.9, which determines the upper bound on the fraction of margin errors and the lower bound of the fraction of support vectors. Four types of kernels are explored: linear, polynomial, radial basis function (rbf), and sigmoid. For polynomial kernels, three degrees are considered: 2, 3, and 4. The *gamma* parameter, which defines the influence of a single training example, is set to either *scale* or *auto*, with additional specific values of 0.001, 0.01, 0.1, and 1.0, providing a range of options for fine-tuning the model's complexity and capability to handle data of varying scales and distributions.

Notably, the average outcomes are generally lower than those achieved in the initial experiment, recording a 0.78 in F1-score.

The results in Table 5.5 demonstrate the performance of ensemble learning strategies (Hard Voting, Hard Bagging, and Stacking) in detecting security requirements for the various datasets combinations. The performance varies across different datasets, even though the Stacking strategy consistently outperforms other strategies in terms of F1-score, demonstrating higher Precision and Recall compared to the other strategies.

#### 5.3.1.1 *Discussion*

The experiment carried above reveals that the most promising ML algorithm is the NuSVC, independently from the WE technique, reaching over 0.80 in Accuracy and F1-score with 3 WE out of 5, highlighting its potential in this task. However, *Lazy Predict* is a valuable tool for quick model evaluation and selection, especially in the early stages of an ML project. This means it should be viewed as a starting point, and more in-depth, customized modeling and feature engineering are often required for optimal performance on real-world datasets. Indeed, we can already observe a substantial difference in the results we get into the *GridSearchCV* experiment, which leads our combination of W2V and NuSVC to earn an average mark of 0.78 in terms of F1-score.

Table 5.3: Results obtained by the built combinations of word-embedder and ML models on the different datasets, for the inter-domain security requirements.
NuSVC = Nu-Support Vector Classification; LR = Linear Regression; RidgeCV = Ridge regression with built-in Cross-Validation; LGBM = Light Gradient Boosting Machine; XGB = eXtreme Gradient Boosting; ET = Extra Trees; SVC = Support Vector Classification.

| **Word2Vec** | | | **BERT** | | | **TF-IDF** | | | **FastText** | | | **GloVe** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | **Acc** | **F1** | **Model** | **Acc** | **F1** | **Model** | **Acc** | **F1** | **Model** | **Acc** | **F1** | **Model** | **Acc** | **F1** |
| **NuSVC** | **0.84** | **0.84** | LGBM | 0.70 | 0.68 | ET | 0.82 | 0.82 | NuSVC | 0.81 | 0.80 | NuSVC | 0.83 | 0.82 |
| LR | 0.82 | 0.83 | XGB | 0.68 | 0.67 | NuSVC | 0.81 | 0.81 | SVC | 0.81 | 0.79 | SVC | 0.81 | 0.80 |
| RidgeCV | 0.82 | 0.82 | ET | 0.69 | 0.67 | RidgeCV | 0.81 | 0.80 | RidgeCV | 0.78 | 0.78 | RidgeCV | 0.80 | 0.80 |

Table 5.4: Results from *GridSearchCV* on the combination Word2Vec and NuSVC.

| **Dataset** | **Precision** | **Recall** | **F1-score** |
|---|---|---|---|
| CPN | 0.86 | 0.85 | 0.84 |
| GPS | 0.85 | 0.82 | 0.83 |
| ePurse | 0.82 | 0.78 | 0.78 |
| CPN & GPS | 0.83 | 0.81 | 0.81 |
| GPS & ePurse | 0.79 | 0.75 | 0.74 |
| CPN & ePurse | 0.73 | 0.72 | 0.71 |
| CPN & GPS & ePurse | 0.80 | 0.78 | 0.78 |
| Overall | 0.81 | 0.79 | 0.78 |

This difference arises because, in the first experiment, the model operates with default hyperparameters. In contrast, *GridSearchCV* explores a predefined set of hyperparameter combinations, which, if not well-defined or excessively broad, may include suboptimal choices affecting model performance. Moreover, the model's performance can vary due to the specific data it's trained on, and *GridSearchCV*, by design, tests the model with diverse data subsets during cross-validation. Consequently, one set of hyperparameters may excel on one fold while another works better on a different one, leading to inconsistent results.

The surprising performance of the Word2Vec and NuSVC combination observed in the *Lazy Predict* library, compared to the traditional *GridSearchCV* implementation, can be attributed to multiple factors. One explanation is that *Lazy Predict* provides an efficient, automated method for evaluating a broad range of ML models with diverse hyperparameters without the need for extensive manual tuning.

This "lazy" approach might serendipitously discover an optimal combination that might not be immediately evident in a structured grid search. Additionally, Word2Vec embeddings capture semantic information, while NuSVC, equipped with a nu-parameter, can effectively handle data with intricate decision boundaries.

Table 5.5: Results from *Ensemble* on the combination Word2Vec and NuSVC, Logistic Regression, and Ridge Classifier CV.

| Dataset | Strategy | Precision | Recall | F1-score |
|---|---|---|---|---|
| CPN | Hard Voting | 0.73 | 0.83 | 0.76 |
| | Hard Bagging | 0.78 | 0.83 | 0.78 |
| | Stacking | 0.92 | 0.90 | 0.90 |
| GPS | Hard Voting | 0.84 | 0.81 | 0.81 |
| | Hard Bagging | 0.79 | 0.78 | 0.77 |
| | Stacking | 0.87 | 0.83 | 0.84 |
| ePurse | Hard Voting | 0.77 | 0.73 | 0.69 |
| | Hard Bagging | 0.69 | 0.71 | 0.66 |
| | Stacking | 0.78 | 0.76 | 0.75 |
| CPN & GPS | Hard Voting | 0.83 | 0.83 | 0.82 |
| | Hard Bagging | 0.83 | 0.83 | 0.82 |
| | Stacking | 0.87 | 0.85 | 0.86 |
| GPS & ePurse | Hard Voting | 0.81 | 0.80 | 0.77 |
| | Hard Bagging | 0.83 | 0.82 | 0.82 |
| | Stacking | 0.82 | 0.77 | 0.77 |
| CPN & ePurse | Hard Voting | 0.79 | 0.78 | 0.77 |
| | Hard Bagging | 0.79 | 0.77 | 0.76 |
| | Stacking | 0.82 | 0.80 | 0.80 |
| CPN & GPS & ePurse | Hard Voting | 0.84 | 0.83 | 0.83 |
| | Hard Bagging | 0.83 | 0.82 | 0.81 |
| | Stacking | 0.83 | 0.81 | 0.82 |
| Overall | Hard Voting | 0.80 | 0.80 | 0.78 |
| | Hard Bagging | 0.82 | 0.80 | 0.79 |
| | **Stacking** | **0.84** | **0.82** | **0.82** |

The *Lazy Predict* library could have identified an ideal configuration during its automated model selection process that aligns well with the unique characteristics of Word2Vec embeddings. Nonetheless, it is crucial to acknowledge that the specific dataset may influence the observed behavior, and results should be interpreted judiciously. Indeed, by looking at Table 5.4, the performances are lower whenever we introduce ePurse requirements, leading the W2V and NuSVC combination never to overcome the 0.8 in F1-score.

For this reason, we search for an ensemble approach of ML algorithms, intending to fill the difficulties of NuSVC to catch security requirements from a particular domain correctly. Of the tested strategies, Stacking is the one that outperforms the others in Precision, Recall, and F1-Score. Even though we do not improve the performance on the specific ePurse dataset, where performance is even weaker than the Grid Search approach, on the 3 combinations of datasets including ePurse, i.e., GPS & ePurse, CPN & ePurse, and CPN & GPS & ePurse, the performance improved of 0.3, 0.9, and 0.4, respectively.

In summary, the results achieved allow to answer our first RQ:

> Stacking is a powerful ensemble learning strategy for intra-domain security requirements detection, offering a balanced trade-off between Precision and Recall. By exploiting Word2Vec, the solution based on stacking NuSVC, Logistic Regression, and Ridge Classifier CV allows to obtain 0.82 in terms of F1-score, with a peak of 0.84 in terms of Precision.

Table 5.6: Results in terms of Precision, Recall, and F1-score achieved with *Ensemble* on the combination Word2Vec and NuSVC, Logistic Regression, and Ridge Classifier CV, compared with the results obtained by Li and Chen [117].

| | **Stacking** | | | **Li and Chen [117]** | | |
|---|---|---|---|---|---|---|
| **Dataset** | **Precision** | **Recall** | **F1-score** | **Precision** | **Recall** | **F1-score** |
| CPN | **0.92** | **0.90** | **0.90** | 0.78 | 0.71 | 0.74 |
| GPS | **0.87** | **0.83** | **0.84** | 0.76 | 0.70 | 0.73 |
| ePurse | 0.78 | 0.76 | 0.75 | **0.90** | **0.80** | **0.85** |
| CPN & GPS | **0.87** | **0.85** | **0.86** | 0.72 | 0.72 | 0.72 |
| GPS & ePurse | 0.82 | 0.77 | 0.77 | **0.85** | **0.80** | **0.83** |
| CPN & ePurse | **0.82** | **0.80** | **0.80** | 0.82 | 0.76 | 0.79 |
| CPN & GPS & ePurse | **0.83** | **0.81** | **0.82** | 0.80 | 0.80 | 0.80 |
| Overall | **0.84** | **0.82** | **0.82** | 0.80 | 0.76 | 0.78 |

Table 5.6 compares the performance of two different approaches: one approach is an ensemble method combining Word2Vec with

NuSVC, Logistic Regression, and Ridge Classifier CV (referred to as "Stacking" in the table). The other is a method reported by Li and Chen [117], as referenced in the table. We select this as the baseline for comparison because the authors achieved the best performance in identifying security requirements in these experiments.

Considering the overall performances, the ensemble approach ("Stacking") outperforms the method by *Li et al.* across all three metrics. The ensemble has an overall Precision of 0.84, Recall of 0.82, and F1-score of 0.82, which are higher than *Li et al.*'s Precision of 0.80, Recall of 0.76, and F1-score of 0.78.

By looking at the performance by datasets, the ensemble method generally achieves better Precision, Recall, and F1-scores than the method by *Li et al.* that performs better on the ePurse and GPS & ePurse datasets. For the ePurse dataset, *Li et al.*'s method has a Precision of 0.90, which is substantially higher than the ensemble's 0.78. Similarly, the Recall is better by 0.04, and the F1-score is better by 0.10. In terms of standard deviation, *Li et al.*'s method shows less variability in performance across different datasets. For example, its F1-scores range from 0.73 to 0.85. In contrast, the ensemble method varies more, with F1-scores ranging from 0.75 to 0.90, indicating that it may be more sensitive to the characteristics of the specific datasets.

In summary, the ensemble method generally shows superior results, especially in overall metrics, while the method by *Li et al.* outperforms the ensemble when using the ePurse dataset or its combination with GPS, suggesting that the ontology-based solution still provides better performance in singular cases.

### 5.3.2 $\boldsymbol{RQ_{Sec}2}$ *How effectively can shallow machine learning algorithms, based on word embeddings, identify security-related requirements coming from different domains?*

Table 5.7 displays the best results achieved by the 150 combinations of WE and ML models in the inter-domain experiment to address $RQ_{Sec}2$. Word2Vec (W2V) as the WE technique and Passive Aggressive Classifier (PAC) as the ML algorithm delivered the highest performance, achieving an Accuracy and F1-Score of 0.66.

Then, we conduct detailed experiments through *GridSearchCV* to identify the optimal hyperparameters for the top combination, i.e., Word2Vec and PAC, and we explore ensemble learning techniques to address specific dataset challenges by leveraging the knowledge of other combination.

Table 5.7: Results obtained by the built combinations of word-embedder and ML models on the different datasets, for the intra-domain security requirements. PAC = Passive Aggressive Classifier; LSVC = Linear SVC; LP = Label Propagation; LS = Label Spreading; LDA = Linear Discriminant Analysis; QDA = Quadratic Discriminant Analysis

| **Word2Vec** | | | **BERT** | | | **TF-IDF** | | | **FastText** | | | **GloVe** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | **Acc** | **F1** | **Model** | **Acc** | **F1** | **Model** | **Acc** | **F1** | **Model** | **Acc** | **F1** | **Model** | **Acc** | **F1** |
| **PAC** | **0.66** | **0.66** | LP | 0.62 | 0.60 | LDA | 0.52 | 0.52 | Perceptron | 0.57 | 0.56 | LR | 0.61 | 0.60 |
| Perceptron | 0.65 | 0.65 | LS | 0.62 | 0.60 | QDA | 0.55 | 0.52 | PAC | 0.56 | 0.56 | PAC | 0.59 | 0.60 |
| LSVC | 0.65 | 0.65 | Bagging | 0.54 | 0.54 | Ridge | 0.51 | 0.51 | LSVC | 0.56 | 0.55 | Perceptron | 0.59 | 0.60 |

Table 5.8: Results from *GridSearchCV* on the combination Word2Vec and PAC.

| **Train set (Test set)** | **Precision** | **Recall** | **F1-score** |
|---|---|---|---|
| CPN (GPS) | 0.72 | 0.72 | 0.70 |
| CPN (ePurse) | 0.71 | 0.44 | 0.39 |
| ePurse (CPN) | 0.66 | 0.61 | 0.63 |
| ePurse (GPS) | 0.77 | 0.51 | 0.46 |
| GPS (ePurse) | 0.72 | 0.73 | 0.71 |
| GPS (CPN) | 0.81 | 0.63 | 0.66 |
| CPN & ePurse (GPS) | 0.74 | 0.44 | 0.34 |
| CPN & GPS (ePurse) | 0.71 | 0.60 | 0.61 |
| GPS & ePurse (CPN) | 0.76 | 0.71 | 0.73 |
| **Overall** | **0.73** | **0.64** | **0.64** |

Table 5.9: Results from *Ensemble* on the combination Word2Vec and PAC, Linear SVC, and Perceptron.

| Train set (Test set) | Strategy | Precision | Recall | F1-score |
|---|---|---|---|---|
| CPN (GPS) | Hard Voting | 0.67 | 0.68 | 0.61 |
| | Stacking | 0.71 | 0.68 | 0.61 |
| CPN (ePurse) | Hard Voting | 0.68 | 0.43 | 0.37 |
| | Stacking | 0.44 | 0.33 | 0.19 |
| ePurse (CPN) | Hard Voting | 0.66 | 0.45 | 0.50 |
| | Stacking | 0.04 | 0.21 | 0.07 |
| ePurse (GPS) | Hard Voting | 0.76 | 0.65 | 0.55 |
| | Stacking | 0.64 | 0.53 | 0.49 |
| GPS (ePurse) | Hard Voting | 0.71 | 0.62 | 0.63 |
| | Stacking | 0.69 | 0.70 | 0.69 |
| GPS (CPN) | Hard Voting | 0.80 | 0.69 | 0.71 |
| | Stacking | 0.80 | 0.77 | 0.78 |
| CPN & ePurse (GPS) | Hard Voting | 0.77 | 0.77 | 0.77 |
| | Stacking | 0.41 | 0.64 | 0.50 |
| CPN & GPS (ePurse) | Hard Voting | 0.74 | 0.53 | 0.52 |
| | Stacking | 0.74 | 0.52 | 0.51 |
| GPS & ePurse (CPN) | Hard Voting | 0.79 | 0.63 | 0.66 |
| | Stacking | 0.60 | 0.48 | 0.53 |
| Overall | Hard Voting | 0.72 | 0.60 | 0.60 |
| | Stacking | 0.61 | 0.56 | 0.56 |

Table 5.8 displays the results obtained from *GridSearchCV* for the Word2Vec and passive aggressive combination. In our experiment, the parameter grid for model tuning consists of several hyperparameters aimed at optimizing the performance of PAC. The regularization parameter $C$ is evaluated at three levels: 0.1, 1.0, and 10.0, to adjust the strength of regularization and prevent overfitting. The maximum number of iterations to run the optimization algorithm is set at 100, 500, and 1000 to control computational intensity and convergence. The tolerance for stopping criteria is tested at three fine granularities: 0.001, 0.0001, and 0.00001, to determine the precision of the solution. Three fractions of the training data, 0.1, 0.2, and 0.3, are considered for validation to aid early stopping, which helps prevent overfitting by halting training when validation scores do not improve. Two types of loss functions, *hinge* and *squared hinge*, are used to see their

impact on the learning algorithm. Finally, the grid includes three values for *n_iter_no_change*: 5, 10, and 20, to specify the number of consecutive iterations without improvement after which training will stop, optimizing computational resources and potentially improving generalization. Even though the performance are lower with respect to the first experiment (the same happen for $RQ_{Sec}1$), the combination still present interesting results, achieving 0.64 in F1-score.

Results in Table 5.9 demonstrate the performance of ensemble learning strategies (Hard Voting and Stacking) in this task. We do not evaluate the hard bagging because it is not possible to set up it with PAC and Perceptron, since these ML solutions do not provide a method to predict probabilities, affecting their adaptability in more complex ensemble scenarios that might require such functionality. We can observe that the performance varies across datasets, with Hard Voting outperforming the other strategies reaching 0.72 in Precision, but presenting lower performances in terms of Recall and F1-score.

#### 5.3.2.1 *Discussion*

The results show that Word2Vec paired with the PAC outperforms other combinations, achieving the highest accuracy and F1-score of 0.66. Other embeddings such as BERT embeddings, which are expected to be strong given their deep contextual representation, do not outperform Word2Vec when used with the models in the experiment, achieving an Accuracy and F1-score up to 0.62 only.

FastText and GloVe embeddings show lower performance in combination with the tested ML models. This suggests that for the specific task of identifying security-related requirements, these embeddings are less effective than Word2Vec in the tested combinations. Traditional frequency-based methods like TF-IDF lag behind sophisticated embeddings, suggesting that the complexity of security-related language requires more nuanced semantic representations.

Linear models like PAC, Perceptron, and LSVC are notable for their robust performance across different embeddings, potentially indicating the presence of linearly separable features in the domain.

From Table 5.8, which presents the results from a *GridSearchCV* experiment using Word2Vec embeddings with a PAC, we can observe

varying levels of generalization capability when models trained on one dataset are tested on another. For instance, the model trained on CPN and tested on GPS performs reasonably well with an F1-score of 0.70, while the same model tested on ePurse drops significantly in performance, with an F1-score of 0.39.

Models trained on combined datasets (e.g., CPN & ePurse, CPN & GPS) do not necessarily outperform those trained on single datasets when tested on a third dataset. For example, the combination of CPN & ePurse tested on GPS shows an F1-score of 0.34, which is lower than the single dataset scenario of CPN (GPS) with an F1-score of 0.70. Considering all the cases, the overall Precision is 0.73, the Recall is 0.64, and the F1-score is 0.64. This indicates that the approach has a moderate level of accuracy and a relatively consistent balance between Precision and Recall. However, the overall F1-score points to limitations in the model's ability to generalize across different domains. Notably, certain dataset pairs like GPS (ePurse) and GPS & ePurse (CPN) exhibit better generalization, with F1-scores of 0.71 and 0.73, respectively, indicating shared or similar characteristics between these domains that the models are able to latch onto. The best individual performance comes from the model trained on GPS & ePurse and tested on CPN, with Precision, Recall, and F1-score of 0.76, 0.71, and 0.73, respectively. This could imply that GPS domain contains features that are more universally applicable or that the CPN test set has characteristics that are easier to predict.

Furthermore, in Table 5.9 we look at the performance of ensemble techniques using a combination of Word2Vec with PAC, Linear SVC, and Perceptron. The combination of CPN for training and GPS for testing shows similar F1-scores for both Hard Voting and Stacking strategies, even though Stacking has a slightly higher Precision but equal Recall. For CPN (ePurse), Hard Voting substantially outperforms Stacking on all metrics. Stacking's F1-score of 0.19 suggests significant issues with generalizing from CPN to ePurse. ePurse (CPN) results are notably poor for the Stacking strategy, with precision falling to 0.04, indicating that the Stacking model is particularly challenged when applying ePurse-trained models to the CPN domain. The ePurse (GPS) and GPS (ePurse) combinations also reflect better performance with Hard Voting, with a notable drop in F1-

score for Stacking when ePurse is the training set. A notable result is the GPS (CPN) combination, where Stacking equals Hard Voting in Precision and outperforms in Recall and F1-score. When combining CPN and ePurse as a training set for the GPS test set, Hard Voting is significantly more effective than Stacking, reflected by a F1-score difference of 0.27. For the combined training sets (CPN & GPS for ePurse, and GPS & ePurse for CPN), Hard Voting consistently outperforms Stacking in terms of F1-score. On aggregate, Hard Voting has higher Precision, Recall, and F1-score than Stacking, with 0.72, 0.60, and 0.60 respectively, compared to 0.61, 0.56, and 0.56 for Stacking. This suggests that, in general, Hard Voting is a more reliable ensemble strategy for cross-domain security requirement identification and that there are significant challenges when models trained on one dataset are tested on another, particularly with the Stacking strategy. This might indicate that the features or patterns learned are not sufficiently universal or are too specific to the training domain.

The data from Tables 5.8 and 5.9 show that performance significantly fluctuates when models are applied to domains they were not trained on, indicating challenges in generalizability. Ensemble methods, including Hard Voting and Stacking, do not consistently outperform the individual classifiers. In particular, Stacking and Hard Voting often result in lower overall performance metrics compared to 0.64 from the single classifier approach. The composition of the training set plays a critical role in the effectiveness of the model on the test set, yet the addition of multiple domains into the training set does not guarantee improved results. For example, models trained on CPN & ePurse and tested on GPS show a dip in F1-score when using a stacking ensemble approach. In conclusion, while there are instances where the models trained on one domain can reasonably perform in another, the overall results suggest that WE-based shallow ML algorithms struggle with cross-domain identification of security-related requirements, as reflected by the variability and generally moderate to low F1-scores across different domain combinations. Ensemble methods do not always result in performance improvements and may require careful tuning to the specific characteristics of the combined domains to optimize results.

In summary, the results achieved allow to answer our second RQ:

Optimizing the hyperparameters for the Word2Vec and Passive Aggressive classifier combination is the most effective approach for detecting inter-domain security requirements, striking a peak of 0.73 in Precision and holding a 0.64 as Recall and F1-score.

Table 5.10: Results in terms of Precision, Recall, and F1-score achieved with *GridSearchCV* on the combination Word2Vec and PAC compared with the results obtained by Li and Chen [117]

| | **GridSearchCV W2V & PAC** | | | **Li and Chen [117]** | | |
|---|---|---|---|---|---|---|
| **Dataset** | **Precision** | **Recall** | **F1-score** | **Precision** | **Recall** | **F1-score** |
| CPN (GPS) | **0.72** | **0.72** | **0.70** | 0.57 | 0.56 | 0.57 |
| CPN (ePurse) | **0.71** | 0.44 | 0.39 | 0.67 | **0.60** | **0.63** |
| ePurse (CPN) | 0.66 | **0.61** | 0.63 | **0.98** | 0.49 | **0.66** |
| ePurse (GPS) | 0.77 | **0.51** | **0.46** | **0.85** | 0.17 | 0.29 |
| GPS (ePurse) | 0.72 | 0.73 | 0.71 | **0.97** | **0.76** | **0.85** |
| GPS (CPN) | **0.81** | 0.63 | 0.66 | 0.66 | **0.76** | **0.70** |
| CPN & ePurse (GPS) | **0.74** | 0.44 | 0.34 | 0.54 | **0.48** | **0.50** |
| CPN & GPS (ePurse) | **0.71** | 0.60 | 0.61 | 0.57 | **0.71** | **0.63** |
| GPS & ePurse (CPN) | 0.76 | 0.71 | 0.73 | **0.98** | **0.73** | **0.84** |
| Overall | 0.73 | **0.64** | **0.64** | **0.75** | 0.58 | 0.63 |

Table 5.10 compares the performance using GridSearchCV with Word2Vec and PAC (W2V & PAC) against the results obtained by Li and Chen [117]. W2V & PAC shows stronger performance in terms of Precision in most dataset combinations compared to Li et al.'s method. This is particularly evident in CPN (GPS), CPN (ePurse), and CPN & ePurse (GPS). In terms of Recall and F1-score, Li et al.'s method outperforms W2V & PAC in several combinations, such as CPN (ePurse), ePurse (CPN), GPS (ePurse), and CPN & GPS (ePurse). This suggests that while the method W2V & PAC may be more precise in its predictions, Li et al.'s method might be better at capturing relevant cases (higher Recall).

Nevertheless, both methods exhibit variability across different dataset combinations. The method W2V & PAC generally maintains a balanced trade-off between Precision and Recall across different

dataset combinations, as indicated by its performance metrics. Notably, the method W2V & PAC has an overall precision of 0.73 and F1-score of 0.64, which is comparable to the F1-score of Li et al. (0.63).

However, Li et al.'s method has a slightly higher overall Precision (0.75) but lower Recall (0.58), that is in contrast with the analysis reported before, in which our combination seems to have a better Precision (5 case out of 9) but worse Recall (6 case out of 9) when compared with Li and Chen [117].

In conclusion, although the W2V & PAC method demonstrates strong Precision in many cases, *Li et al.*'s method occasionally surpasses it in Recall and F1-score. This highlights the complexity and context-dependent nature of requirements, influencing the selection of the suitable approach for inter-domain identification of security-related requirements, despite the results favoring W2V & PAC.

### 5.3.3 $\boldsymbol{RQ_{Sec}3}$ *How effective can pre-training context in BERT transformers be on the detection of security-related requirements?*

Table 5.11 presents the results from different pre-training setups of BERT models to detect security requirements. The results include Precision, Recall, and F1-score for different models trained on specific datasets and tested on both $\text{Test}_{Data}$ and the $\text{Promise}_{exp}$ dataset. The models evaluated are BERT, CweCveBERT, CodeBERT, and CweCveCodeBERT, already detailed in Section 5.2.1.

Table 5.11: Results from different pre-training setups of BERT to detect security requirements.

| Model | Test set | Precision | Recall | F1-score |
|---|---|---|---|---|
| BERT | **$\text{Test}_{Data}$** | **0.62** | **0.79** | **0.69** |
| | $\text{Promise}_{exp}$ | 0.69 | 0.71 | 0.70 |
| CweCveBERT | $\text{Test}_{Data}$ | 0.44 | 0.68 | 0.53 |
| | $\text{Promise}_{exp}$ | 0.69 | 0.56 | 0.62 |
| CodeBERT | $\text{Test}_{Data}$ | 0.59 | 0.72 | 0.65 |
| | **$\text{Promise}_{exp}$** | **0.75** | **0.76** | **0.76** |
| CweCveCodeBERT | $\text{Test}_{Data}$ | 0.50 | 0.72 | 0.59 |
| | **$\text{Promise}_{exp}$** | **0.69** | **0.82** | **0.75** |

For BERT, testing on the $\text{Promise}_{exp}$ dataset yields similar results in terms of Precision, Recall, and F1-score compared to the results on $\text{Test}_{Data}$ dataset. In comparison, CweCveCodeBERT and CodeBERT exhibit strong performance when tested on $\text{Promise}_{exp}$, but lower on $\text{Test}_{Data}$. Table 5.12 presents the inference performances of pre-trained BERT models on industrial specifications. The models evaluated are BERT, CodeBERT, and CweCveCodeBERT, and their performance metrics include Precision, Recall, and F1-score for various test sets (CPN, ePurse, GPS, and average across all sets).

Table 5.12: Inference performances of pre-trained BERT models on the industrial specifications.

| | BERT | | | CodeBERT | | | CweCveCodeBERT | | |
|---|---|---|---|---|---|---|---|---|---|
| **Test set** | **Precision** | **Recall** | **F1-score** | **Precision** | **Recall** | **F1-score** | **Precision** | **Recall** | **F1-score** |
| CPN | 0.76 | 0.77 | 0.76 | 0.78 | 0.78 | 0.78 | **0.80** | **0.78** | **0.79** |
| ePurse | 0.81 | 0.53 | 0.50 | 0.82 | 0.61 | 0.61 | **0.82** | **0.62** | **0.62** |
| GPS | 0.79 | 0.76 | 0.76 | 0.78 | 0.76 | 0.76 | **0.79** | **0.76** | **0.76** |
| Average | 0.79 | 0.69 | 0.67 | 0.79 | 0.72 | 0.72 | **0.80** | **0.72** | **0.72** |

Table 5.13: McNemar test results.

| **Dataset** | **BERT vs CodeBERT** | **BERT vs CweCveCodeBERT** | **CodeBERT vs CweCveCodeBERT** |
|---|---|---|---|
| CPN | 0.228 | **0.006** | 0.096 |
| ePurse | 0.034 | **0.010** | 1 |
| GPS | 0.724 | 1 | 0.683 |

CweCveCodeBERT and CodeBERT consistently outperform BERT on all test sets out of GPS, achieving higher Precision, Recall, and F1-score. The average performance across all test sets is best for CweCveCodeBERT, even though there is not that big difference with respect to CodeBERT. Table 5.13 presents the results of the McNemar test comparing the performance of BERT, CodeBERT, and CweCveCodeBERT on the CPN, ePurse, and GPS datasets. In the comparison between BERT and CodeBERT, CweCveCodeBERT shows a significant improvement on the CPN and ePurse dataset with a very low p-value, indicating its superior performance. As expected, for the GPS dataset, CweCveCodeBERT performs on par with CodeBERT, suggesting that both models are effective in this context.

#### 5.3.3.1 *Discussion*

Our analysis reveal that standard BERT demonstrated robust performance, achieving the highest F1-score of 0.69 on $\text{Train}_{Data}$ ($\text{Test}_{Data}$) and maintaining a similar F1-score of 0.70 on $\text{Promise}_{exp}$. This consistency suggests that BERT possesses a commendable ability to generalize across diverse contexts. CweCveBERT, tailored for CWE and CVE data, exhibited a lower F1-score of 0.53 on $\text{Test}_{Data}$, slightly improving to 0.62 on $\text{Promise}_{exp}$, implying a potential limitation in its adaptability to varying data domains. CodeBERT, with its emphasis on code-centered pre-training, shows moderate performance on $\text{Test}_{Data}$ (F1-score of 0.65) but excellent on $\text{Promise}_{exp}$ with an F1-score of 0.76.

This superior performance underlines the relevance of its pre-training context for the $\text{Promise}_{exp}$ dataset. CweCveCodeBERT, amalgamating features of both CWE/CVE and code-related data, shows balanced efficacy across $\text{Test}_{Data}$ and $\text{Promise}_{exp}$, reaching an F1-score of 0.75 on the latter. This indicates that hybrid pre-training approach can yield models with broader applicability for security detection.

Following these findings, we extended our examination to the inference capabilities of the top-performing BERT models (BERT, CodeBERT, and CweCveCodeBERT) across diverse industrial datasets (CPN, ePurse, GPS), as presented in Table 5.12. CweCveCodeBERT consistently outperformed the others, especially in Precision and F1-score, across all datasets. Its superior performance, particularly on the CPN dataset, suggests an effective alignment of its pre-training context with the dataset's characteristics. However, a noticeable decrease in Recall was observed across all models on the ePurse dataset, implying a potential challenge in capturing all relevant cases. The GPS dataset saw similar performance levels across all models, indicating a uniform handling of the dataset's features by the models. While CweCveCodeBERT slightly led in overall metrics, the marginal difference compared to CodeBERT and BERT indicates a general effectiveness of all three models for these industrial specifications.

To rigorously assess the statistical significance of the performance differences among these models, we applied the McNemar test with a Bonferroni correction, as shown in Table 5.13.

This correction was crucial to mitigate Type I errors due to multiple comparisons. The test results suggest significant differences between BERT and CweCveCodeBERT on the CPN and ePurse datasets, underscoring CweCveCodeBERT's superior performance. However, no statistically significant differences can be observed between CodeBERT and CweCveCodeBERT across the datasets, suggesting comparable effectiveness. The significant difference between BERT and CodeBERT on ePurse, with a p-value of 0.034, calls for cautious interpretation due to its proximity to the adjusted threshold.

In summary, the results achieved allow to answer our $RQ_{Sec}3$:

> The introduction of CodeBERT and CweCveCodeBERT, with their unique pretraining on code and/or security descriptions, showcase the models's superior performance and domain independence, marking, in the case of CweCveCodeBERT, a significant advancement in identifying security-related requirements when compared to the standard BERT model.

Table 5.14: Results in terms of Precision, Recall, and F1-score achieved with *CweCveCodeBERT* compared with the results of Mohamad et al. [151] and Munaiah, Meneely, and Murukannaiah [154]

| | CweCveCodeBERT | | | Mohamad et al. [151] | | | Munaiah, Meneely, and Murukannaiah [154] | | |
|---|---|---|---|---|---|---|---|---|---|
| **Dataset** | **Precision** | **Recall** | **F1-score** | **Precision** | **Recall** | **F1-score** | **Precision** | **Recall** | **F1-score** |
| CPN | **0.80** | 0.78 | **0.79** | 0.53 | 0.71 | 0.61 | 0.73 | **0.80** | 0.74 |
| ePurse | 0.82 | 0.62 | 0.62 | **0.95** | **0.68** | **0.79** | 0.61 | 0.65 | 0.61 |
| GPS | **0.79** | 0.76 | **0.76** | 0.63 | **0.81** | 0.71 | 0.68 | 0.67 | 0.68 |
| Average | **0.80** | 0.72 | **0.72** | 0.70 | **0.73** | 0.70 | 0.67 | 0.71 | 0.68 |

Table 5.14 compares the performance of CweCveCodeBERT with two other models from previous researches (Mohamad's approach using TF-IDF, SMOTE, and Random Forest, and Munaiah's approach using a One-Class SVM trained on CWE descriptions) in terms of Precision, Recall, and F1-score across various datasets. CweCveCodeBERT generally shows the highest average Precision and F1-score across all datasets, indicating its strong ability to correctly identify and confirm relevant cases.

However, it has a slightly lower average Recall compared to Mohamad's approach, suggesting it may miss some relevant cases that Mohamad's model catches.

In particular, on the CPN dataset, CweCveCodeBERT outperforms the other models in terms of Precision and F1-score, but Munaiah's model has a higher Recall. For the ePurse dataset, Mohamad's approach shows significantly better performance, suggesting that for this particular dataset, their method might be more effective. On the GPS dataset, CweCveCodeBERT leads in terms of Precision and F1-score, while presenting a lower Recall compared to Mohamad's solution. Munaiah's One-Class SVM model shows moderate performance across all metrics. It appears to be more balanced but does not excel in any specific area compared to the other models.

Clearly, implementing CweCveCodeBERT may require more computational resources and expertise in BERT models and NLP, but it offers a high degree of Precision and overall F1-score, making it a strong candidate for practical applications where accuracy is paramount. Mohamad's approach, while less precise, might be easier to implement due to the traditional ML pipeline involving TF-IDF and Random Forest.

It could be suitable for scenarios where maximizing Recall is more important, but this implies the availability of numerous requirements to obtain the importance of security-related terms with TF-IDF. Munaiah's One-Class SVM approach might offer a simpler implementation with moderate performance, potentially serving well in scenarios where a balance between precision and recall is needed.

Ultimately, CweCveCodeBERT's robust performance across diverse datasets underscores its suitability as an inter-domain model. Its capacity to adapt to various contexts while maintaining high Precision and F1-scores highlights its versatility, making it an effective tool for detecting security-related requirements without dependence on specific application domains.

### 5.3.4 *Implications*

As we delve into the intricacies of our research findings, it becomes increasingly evident that the realm of security requirement detection in software engineering is multifaceted, with varying approaches suited to different scenarios and resource availabilities.

In addressing $RQ_{Sec}1$, our investigation sheds light on the applicability and effectiveness of ensemble methods for engineers working within established domains. These methods, known for their minimal resource and effort demands, exhibit a notable capability in identifying whether new requirements in ongoing projects pertain to security, offering a streamlined, resource efficient approach for projects with well defined security requirements, allowing for swift and effective classification within known parameters. For researchers, the success of embedding techniques such as Word2Vec and GloVe in recognizing security requirements suggests the need for further investigation into which features of these embeddings most contribute to classification accuracy. Additionally, studying dataset characteristics that influence model performance could lead to the development of more robust and adaptable models.

For practitioners, the effective combinations of embedding techniques and ML algorithms identified can be directly applied in development environments to automate security requirement identification, thus saving time and reducing errors in manual processes. This approach allows practitioners to tailor ML methods to the specific characteristics of their project's domain, enhancing the precision of requirement detection and integrating these ML techniques into tools to maintain high security in software projects.

Conversely, $RQ_{Sec}2$ findings reveal a contrasting landscape. Here, practitioners tasked with understanding the security implications of requirements for new projects, especially those diverging from previously encountered domains, must navigate more complex waters. Customized solutions become paramount in these scenarios, as the characteristics and nuances of security requirements can significantly vary across different domains. Such an approach necessitates a deeper, more tailored analysis, acknowledging the distinctiveness of each domain.

Thus, practitioners should be cautious in using models trained on specific domain data without adjustments and evaluate the domain adaptability of each model before deployment. The development of domain-agnostic tools that leverage adaptable models could reduce the need for multiple specialized tools, ensuring broader applicability. On the other hand, researchers are encouraged to explore models that can adapt to varying contexts without retraining, potentially through meta-learning approaches or transfer learning strategies to improve generalization capabilities.

$RQ_{Sec}3$ brings us to the cutting edge of our exploration, showcasing the potential of transformer architectures in building inter-domain models. For practitioners equipped with adequate resources, these models emerge as powerful tools that surpass the constraints outlined in $RQ_{Sec}2$. By leveraging the advanced capabilities of transformer architectures, practitioners can develop models that maintain high accuracy and adaptability across various domains, mitigating the need for inter-domain customization.

Moreover, integrating pre-trained BERT models into development pipelines could enhance the detection and categorization of security requirements, improving the early phases of software development and ensuring compliance with security standards. These models' adaptability to specific security contexts could also be used to develop customized solutions for industries with stringent security needs, such as finance and healthcare. For the researchers, the success of pre-trained BERT models in inter-domain tuning opens up new research avenues, including the fine-tuning of language models for specialized applications that could extend beyond security to other types of non-functional requirements. Further comparative analysis of different pre-training corpora and their impacts on model performance in specific application contexts could be beneficial.

In conclusion, for the research community, our findings offer a foundational step towards achieving domain independence in detecting security requirements. The pursuit of this independence is challenging, thus the question:

**To what extent can we truly achieve domain-agnostic models in the field of security requirement detection?**

This inquiry not only opens new avenues for research but also beckons a deeper understanding of the interplay between machine learning techniques and the multifarious nature of security requirements across diverse domains.

## 5.4 THREATS TO VALIDITY

This section elaborates on the threats to the validity of the performed study, which stem from the generalizability and repeatability of the presented results and the correctness of the used tools.

**Construct Validity**. It involves determining how well a test measures the concept it evaluates by checking the adequacy of observations and inferences based on the measurements performed during the study [28]. It is critical in establishing the overall validity of a method. In our context, methods offered by the Scikit-learn library have been used for the evaluation. In particular, we used the `accuracy_score`[6] method to measure Accuracy, the `precision_score`[7] method to measure Precision, the `recall_score`[8] method to measure Recall, and the `f1_score`[9] method to measure F1-score.

As much as relying on the results of a single tool may pose a threat to validity, especially in the case of deep learning applications, since these metrics can be calculated mathematically, they should not be tool-dependent. Furthermore, we apply these metrics because they have been used in previous studies that analyzed approaches to detect security requirements, and this allows the results obtained here to be compared with those obtained in previous studies.

**Internal Validity**. It refers to the validity of results by considering causality between action taken and the resulting change that can be observed [28]. In our study for exploring the applicability of our models to detect security-related aspects, we assume that these

6 https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html

7 https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html

8 https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html

9 https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

models are comparable to each other, as they come from the same libraries, i.e., Scikit-learn and Transformers[10]. Causality could be a threat to the internal validity of our study. However, the statistical test results show that the measurements were significant, implying that the correlations found derive from fairly strong causal relationships and reinforcing the idea that the conditions for causality in the approach are met.

**External Validity**. It concerns the generalizability and repeatability of the produced results [28] and, therefore, the usefulness of the results of a research study, i.e., is the researchers' results applicable in a real-world context?

Our approach is based on Python since the developed models have been implemented with the libraries available in this programming language. As far as we know, the library can be used only in Python, so the reproducibility of the BERT-based models in other programming languages has not been studied in terms of feasibility. To demonstrate the generalizability of our work, we apply a k-fold cross-validation technique where possible, showing that the approach does not depend on the data used for training. In addition, the datasets are software specifications from different application domains, which corroborates the result of our study, demonstrating applicability in industrial environments. To promote the replication of this work, we made all the tools, scripts, and data available [6].

**Conclusion Validity**. It refers to the degree of reasonableness or correctness of conclusions regarding the defined null hypothesis and the performed statistical tests [28]. We consider McNemar's test to compare the performances of two models because it has a lower probability of error and makes fewer assumptions. When declaring that one model performs better than another, we consider accepting a probability of 5% of committing a Type-I-Error [232]. Thus, when null hypotheses are rejected, we consider the relationship between data and result to be reasonable.

10 https://huggingface.co/docs/transformers/main/en/index#transformers

# 6

# A FIRST EYE ON THE IMPACT OF QUANTUM NATURAL LANGUAGE REPRESENTATIONS FOR NON-FUNCTIONAL REQUIREMENTS CLASSIFICATION

## 6.1 INTRODUCTION

Non-Functional Requirements (NFRs) represent a critical aspect of Software Engineering (SE), providing specifications related to the system's quality attributes, such as security, performance, and usability. Unlike functional requirements, which describe what the system should do, NFRs define how well the system performs its functions [5, 21]. Accurately identifying and classifying NFRs early in the software development life cycle is crucial, as failure to do so can lead to cost overruns, project delays, and system failures [162, 228]. Consequently, automatic classification of NFRs has become a focal point in recent research efforts [75, 100, 136].

Recent advancements in ML and NLP have provided new approaches for automating the classification of NFRs. Shallow ML models, such as Support Vector Machines (SVM) and Random Forests (RF), combined with word embedding techniques like Word2Vec [146], GloVe [169], and FastText [20], have demonstrated significant success in the NFR classification task [10], even though they are combined with Deep Learning (DL) solutions, like neural networks or even advanced approaches like Transformers [99, 220]. Despite the promising results achieved by ML, DL and NLP models, these approaches show limitations when capturing complex dependencies among words or when dealing with a low amount of data.

In the meanwhile, Quantum Natural Language Processing (QNLP) represents an emerging frontier in computational linguistics and artificial intelligence [74]. This innovative paradigm leverages the principles of quantum mechanics, such as superposition and entanglement, to encode and process linguistic structures more efficiently

than classical methods. In QNLP, grammatical structures and the semantics of words are modeled as tensor networks or quantum circuits, opening up new possibilities for enhancing NLP tasks [49, 231]. QNLP models, such as those implemented using the Distributional Compositional Categorical (DisCoCat) framework, allow for the compositional representation of sentence meaning using quantum circuits [96]. These methods have shown theoretical potential for achieving significant computational advantages, particularly in tasks requiring the representation of high-dimensional and complex linguistic structures [1, 78, 208].

So, in the first side, classical NLP approaches such as BERT [57] and TF-IDF [182] often struggle with complex linguistic structures due to their reliance on surface-level co-occurrence and statistical patterns. These models may miss deeper compositional elements of language, especially in tasks that demand understanding grammar and semantics. In contrast, QNLP, which uses string diagrams and tensor networks, can map such linguistic structures directly into quantum circuits, providing a more flexible and holistic representation of language. Additionally, classical models tend to be data-demanding, requiring large datasets to perform well, and often suffering when data is limited. QNLP offers advantages by exploiting quantum properties like superposition and entanglement, enabling the expectation for better performance with smaller datasets. By leveraging categorical quantum mechanics, of which we offer an overview in Section 6.2, QNLP can model contextual dependencies within a specific language, positioning it as a promising tool for tasks like NFR classification, where word dependencies are critical.

This work presents an empirical study on the application of QNLP to the automatic detection of NFRs defined as a multiclass classification task, a novel exploration in the software engineering domain. Our work explores three distinct approaches: 1) the use of shallow ML models combined with word embedding techniques, which serve as a baseline for comparison; 2) a basic model based on tensor networks, using string diagrams to represent NFRs; 3) quantum models that transform string diagrams into quantum circuits and simulate quantum computation for the NFR classification task. In particular, despite the state of the art on multiclass classification of NFR offers a

lot of solutions [10, 101, 133], their focus is on the best performing solution based on specific techniques and technologies. In addition, they employ widely used large dataset like Promise [121] and Pure [68], applying different pre-processing and different requirements selection. Thus, our first investigation ($RQ_{Nfr}1$) aims to assess the performance of the shallow ML algorithms like SVM, RF, Linear Regression and so on, when requirements are represented with word embedding techniques. Our approach differs from the literature in preprocessing techniques, data volume, and on the choice of shallow ML solutions rather than advanced DL approaches, like Convolutional Neural Network (CNN) or BERT. This approach opens the way for the exploration of new techniques for representing requirements, namely string diagram. String diagrams provide a visual and compositional way to represent the grammatical structure of sentences, capturing the relationships between words and their meanings through the use of tensor networks. Representing requirements as tensor networks is an unexplored path which we want to bring to light with this work. This guided our second investigation ($RQ_{Nfr}2$), focusing on the impact of string diagrams application in NFRs multi-class classification. Lastly, the employment of string diagrams allow the exploitation of quantum technologies and techniques, as string diagrams can be manipulated and converted to quantum circuits, enabling the simulation of quantum computation for the NFR classification task. This is the focus of our last investigation ($RQ_{Nfr}3$) in which we evaluate the representation of NFR in quantum circuits in different settings simulating quantum environments, carrying out shot-based and non shot-based experimentations.

In response to $RQ_{Nfr}1$, we find that shallow ML algorithms, when combined with word embeddings like GloVe and Word2Vec, produced highly effective models for classifying NFRs. These models achieve precision, recall, and F1-score metrics exceeding 0.80, demonstrating the strength of these traditional statistical embeddings in capturing the semantic relationships in NFRs. However, embeddings such as TF-IDF and BERT, which are based on word frequency and contextual embeddings, struggle in the classification task, with significantly lower F1-scores, highlighting the difficulty of these models in handling the volume of data and the specific preprocessing steps

used in this study. $RQ_{Nfr}2$ explore the potential of tensor network-based models, where requirements were represented through string diagrams. The Cups Reader, which represents language as a sequence of tensors, outperforms the syntax-based Bobcat Parser, particularly when coupled with Cross Entropy loss in a PyTorch model. While the overall performance was not as high as traditional ML techniques, the results were encouraging, showing that string diagram representations can offer a competitive alternative, particularly outperforming BERT-based models in several NFR classes such as Operability, Security, and Usability. This suggests that tensor-based representations of NFRs may offer a novel path forward for requirements classification, though further refinements are necessary to improve their overall efficacy. Finally, in addressing $RQ_{Nfr}3$, our experiments with quantum circuit-based models reveal a more complex landscape. The non-shot-based simulations, which transform quantum circuits into tensor networks using a Numpy model, demonstrate comparable results to those in $RQ_{Nfr}2$, even though with lower performance overall. Meanwhile, the shot-based simulations show that certain NFR classes, such as Operability, could achieve high recall, but the remaining results are generally scarce. The nature of quantum computation and additional factors, such as backend configuration and shot counts, introduce new complexities, making it challenging to fully evaluate the potential of quantum technologies for NFR classification in this initial exploration. Despite this, the possibility to represent NFRs through quantum circuits remains a promising direction, particularly as quantum hardware and simulation environments evolve.

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

1) A benchmark of combinations of word embeddings and shallow ML algorithms that differentiate from the state of the art on NFR classification task;

2) A first attempt to classify NFRs represented as string diagrams using tensor networks.

3) A novel application of quantum computing techniques for the NFR classification task, including the use of quantum circuits for processing linguistic structures.

4) A replication package to replicate and build upon our work [7].

5) A road-map to further investigate the application of QNLP techniques and technologies for NFRs classification task;

## 6.2 INTRODUCTION TO QUANTUM NLP

Quantum NLP is a developing field that applies quantum computing principles to the challenges of NLP. This field aims to leverage the unique capabilities of quantum computers—such as superposition, entanglement, and quantum parallelism—to enhance the performance of NLP tasks, hopefully surpassing the limits of classical computational methods. Traditional NLP models, such as word embeddings (e.g., Word2vec [147]) and deep learning architectures like Transformers [220], have revolutionized the processing of human language. However, these models are inherently bound by the constraints of classical computation. They typically represent words as vectors in a high-dimensional space and apply operations that, while effective, are limited in capturing the full complexity of a language with its ambiguity.

QNLP introduces a new paradigm by representing linguistic elements as quantum states. In this framework, words, phrases, and sentences are encoded into quantum systems, where the principles of quantum mechanics allow for a more expressive representation of linguistic meaning. Superposition enables the simultaneous representation of multiple meanings or interpretations of a word, while entanglement allows for the complex dependencies between words to be captured more naturally than in classical models.

One of the foundational models in QNLP is the Distributional Compositional Categorical (DisCoCat) model, which combines category theory with distributional semantics to model the meaning of sentences in a compositional manner [49]. This model provides a natural fit for quantum computing, as it aligns with the structure of quantum mechanics, where compositionality plays a key role.

To facilitate the development and experimentation of QNLP models, the lambeq library [97] was created by the Quantinuum QNLP team[1]. Lambeq is an open-source Python framework that provides

1 https://www.quantinuum.com/

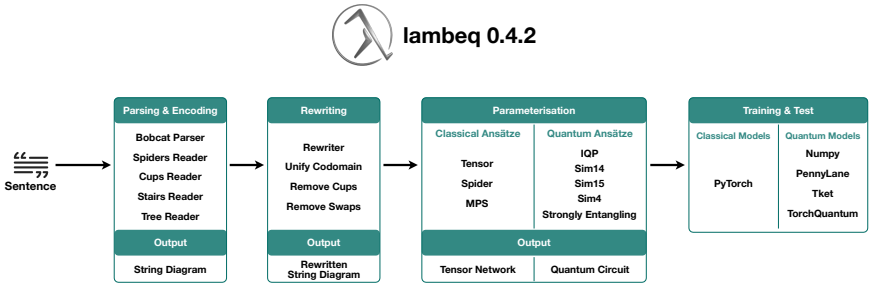


Figure 6.1: The general pipeline implemented through lambeq.

functionality for converting any sentence into a quantum circuit, utilizing a specified compositional model along with particular parameterizations and selected ansätze. The process of converting a sentence into a quantum circuit involves several key steps, each offering a variety of functionalities. These steps include parsing and encoding, rewriting, parameterisation, and training/testing, as depicted in Figure 6.1. We summarize the various features provided from the framework. The full description of each can be found in the specific documentation[2], from which we grasped most of the following information and in which can be found visual example for each kind of parsing, rewriting and parameterisation.

**Parsing & Encoding.** The parsing process is the step that converts a sentence into a string diagram. Lambeq's string diagrams are equipped with types that represent the interactions between words according to the pregroup grammar formalism [112]. When annotated with pregroup types, the diagram for a sentence shows each wire labeled with an atomic type or an adjoint. For instance, in the sentence "John walks in the park", the noun "John" might be labeled with type $n$, corresponding to a noun or noun phrase, while the verb "walks" could be labeled with type $s$, corresponding to a sentence. The adjoints $n^r$ and $n^l$ indicate that a noun is expected to the right or left of a specific word, respectively. The library provides different syntactic models, offering flexibility in how linguistic structures are represented and processed. In the following we detail the

2 https://cqcl-dev.github.io/lambeq/intro.html

various parsing methods supported by lambeq, with a focus on the DisCoCat model and alternative approaches such as bag-of-words, word-sequence models, and tree readers.

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

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

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

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

Tree Reader provide another syntax-based parsing approach, leveraging the structure of CCG derivations. A CCG derivation, which follows a biclosed form, is directly interpreted as a sequence of compositions. By default, composition is handled by a single "cell" named UNIBOX, even though lambeq offers customization through the TreeReaderMode, allowing for different cells based on CCG rules or rule-type pairs and enabling more fine-grained control over the composition process.

**Rewriting.** The rewriting stage is adopted for simplifying and optimizing string diagrams derived from syntactic derivations. These diagrams, which represent the structure and relationships between words in a sentence, can become complex, leading to increased computational costs and longer training times. The rewriting process aims to address these issues by applying various rewriters and rewriting

rules to reduce the complexity of the diagrams. The rewriting can be applied at two stages: box-level and diagram-level.

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

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

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

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

The Remove Swaps Rewriter eliminates swaps from a diagram, producing a proper pregroup diagram that aligns with J. Lambek's definition of pregroup grammars [112]. This rewriter is essential for maintaining the structural integrity of the diagram in accordance with the theoretical underpinnings of pregroup grammar.

**Parameterisation.** The parameterisation phase consists in transforming the string diagrams derived from sentences into concrete quantum circuits or tensor networks. This stage involves the ap-

plication of ansätze, which define the specific parameters for the quantum or classical systems, such as the number of qubits or tensor dimensions associated with each grammatical type in the sentence.

In classical experiments, Tensor, Matrix Product States and Spider Ansatz can be applied to convert the string diagram into a tensor network, assigning dimensions to the atomic types in the diagram, defining how grammatical types are represented as tensors.

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

In quantum experiments, the string diagram is converted into a quantum circuit by applying a Circuit Ansatz. Lambeq provides several quantum ansätze, each with different configurations and properties suitable for various quantum computing tasks. The Instantaneous Quantum Polynomial (IQP) Ansatz interleaves layers of Hadamard gates with diagonal unitaries. The diagonal unitaries are implemented using adjacent controlled-Rz (CRz) gates [76]. This ansatz is particularly useful for creating circuits that are classically hard to simulate. To convert a string diagram into an IQP circuit, you assign qubits to atomic types (e.g., 1 qubit for nouns and 1 qubit for sentences) and specify the number of IQP layers. Sim14 Ansatz is a modification of Circuit 14 define by Sim et al. [197], where the circuit-block construction is replaced with two rings of CRx gates, oriented in opposite directions. It provides a different entanglement structure compared to the IQPAnsatz. Sim15 Ansatz is similar to Sim14Ansatz, but uses two rings of CNOT gates instead of CRx gates, providing

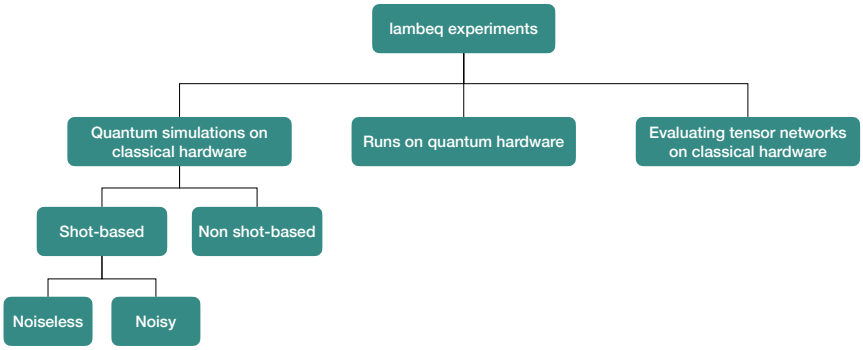


Figure 6.2: Hierarchy of experimental applications in lambeq.

another variation in the entanglement structure. Sim4 Ansatz corresponds to Circuit 4 defined as Circuit 14 by Sim et al. [197]. It employs a layer of Rx gates followed by Rz gates, and a ladder of CRx gates per layer, making it a more straightforward but effective quantum circuit design. Adapted from the PennyLane implementation, Strongly Entangling Ansatz uses three single-qubit rotations (RzRyRz) followed by a ladder of CNOT gates with varying ranges per layer. This ansatz is designed to create strong entanglement across the qubits, which is crucial for certain quantum algorithms.

**Training & Test.** The training and testing stage involves using quantum circuits or tensor networks obtained from the parameterisation to train models for various machine learning tasks. Lambeq provides several models, each suited to either quantum or classical experiments. All the possible experiments that can be done through lambeq are shown in Figure 6.2.

The model employed to evaluate tensor networks on classical hardware is the Pytorch Model. It treats string diagrams as tensor networks, where each box in the diagram corresponds to a tensor, and the edges between boxes define how these tensors are contracted (i.e., how the data flows between different parts of the model). The tensor contractions in this model are optimized using opt_einsum, a library that efficiently handles complex tensor operations. This makes the Pytorch Model capable of handling large and intricate tensor networks typical in advanced machine learning tasks. The

Pytorch Model is versatile and can be integrated with classical neural networks. Users can subclass the model and override the forward method to combine the outputs of tensor diagrams with other neural network layers, allowing the creation of customized architectures.

The Numpy Model is a quantum model that utilizes low-level simulators to convert quantum circuits into tensor networks. These tensor networks are then contracted efficiently using the opt_einsum library. The model supports two types of simulators: unitary simulators and density matrix simulators. The Unitary Simulator is used when the circuit contains only Bra, Ket, and unitary gates. It converts the circuit with n output qubits into a tensor of shape (2, ) * n, which is a pure quantum state. On the other hand, the Density Matrix Simulator is used when the circuit involves operations like Encode, Measure, or Discard, which result in a mixed quantum state. It converts the circuit into a tensor of shape (2, ) * (2 * n + m), where n is the number of output qubits and m is the number of output bits. This allow the exact (non shot-based) simulation of quantum circuits on classical hardware.

The PennyLane Model leverages the capabilities of PennyLane[3], a popular quantum machine learning library, along with PyTorch, to create hybrid quantum-classical models. This model allows to perform classical-quantum machine learning experiments where quantum circuits are combined with classical neural networks. The model offers two modes of operation: State Vector Simulation and Probability Simulation. The former mode simulates the quantum circuit as a pure quantum state using unitary operations. It is efficient for simulation purposes but does not account for measurement noise. The latter mode simulates the quantum circuit as a mixed state, taking into account the probabilities of different outcomes after measurements. This mode is required when running experiments on real quantum hardware, where measurement noise and errors are present. The PennyLane Model can be used to optimize simulated circuits using exact back-propagation in PyTorch, which can lead to more accurate training compared to optimization methods like Simultaneous Perturbation Stochastic Approximation (SPSA) [207].

3 https://pennylane.ai/

However, exact back-propagation is not feasible on real quantum hardware, where the parameter-shift rule is used instead. Additionally, the model supports hybrid architectures where the output of a quantum circuit is fed into a classical neural network. This allows for the creation of complex models that leverage both quantum and classical computational advantages.

The Tket Model is designed for running quantum circuits on real quantum hardware or in simulated environments that account for hardware-specific noise and architecture constraints, allowing quantum simulations in noisy shot-based environment. This model utilizes the "pytket" library, a high-performance simulator framework that is a part of the Qiskit suite, developed by IBM [175], which provides access to various quantum hardware backends. This framework supports a variety of backends, including AerBackend (the one that we used in our experiments) for noisy simulations of quantum machines and direct execution on devices like Quantinuum's H-series or IBM Quantum hardware. The Tket Model is particularly useful for experiments that require shot-based results, which are the counts of measurement outcomes after running a quantum circuit multiple times. These shot-based results are essential for understanding the probabilistic nature of quantum computing. The model can be configured to use different noise models and compilation passes specific to the hardware, making it a versatile choice for experiments that need to consider the limitations and characteristics of quantum devices.

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

## 6.3 EMPIRICAL STUDY DESIGN

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

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

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

### 6.3.1 *Motivations of Research Questions*

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

techniques are suitable for the specific task. This consideration led to the formulation of our first research question:

> **RQ$_{Nfr}$1.** How effectively can shallow machine learning algorithms classify non-functional requirements when represented with word embeddings techniques?

By answering RQ$_{Nfr}$1 we have the baseline for the comparison against Quantum NLP strategies. Since we have different choices in this new approach (see Figure 6.2), as described in Section 6.2, we start our investigation by evaluating tensor networks on classical hardware, exploiting the classical path in the pipeline shown in Figure 6.1. This choice is due to the fact that first and foremost we want to understand how the representation of requirements by string diagrams impacts the performance of the models, without changing the underlying computational model. This also allows us to make a comparison with classical methods (word embeddings and shallow ML) that exploit a different representation of requirements to predict their class. This lead to the definition of the following research question:

> **RQ$_{Nfr}$2.** How effectively can a basic model classify non-functional requirements when represented as string diagrams and parameterised as tensor networks?

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

**RQ$_{Nfr}$3.** How effectively can quantum models classify non-functional requirements when represented as string diagrams and parameterised as quantum circuits?

Quantum computing holds the promise of enhanced problem-solving capabilities in certain computational tasks, and this research question aims to understand whether quantum circuits, representing language as tensors, can lead to better or comparable classification performance to classical machine learning and tensor-based models. Given the probabilistic nature and complex configurations involved in quantum computing (e.g., shots, compilation passes), RQ$_{Nfr}$3 investigates whether quantum-enhanced models can provide a tangible advantage in this domain or whether they introduce new challenges, such as parameterization and circuit complexity. Given the unavailability of quantum resources, the only option available to us is to simulate a quantum environment, which, as shown in Figure 6.2, can be non-shot based or shot based. Therefore, we explored both avenues, leaving experimentation on quantum resources as possible future research.

### 6.3.2 *Dataset*

To answer the research questions presented in the previous Section 6.3.1, we exploit the PROMISE expanded dataset [121]. We choose this dataset because previous studies exploited it [177, 179, 193] to deal with the NFRs classification task.

It includes in total 969 requirements detailed in Table 6.1 coming from 15 different projects. However, we do not use the entire dataset because the conversions of requirements into string diagrams and, then, into tensor networks or quantum circuits could be hard to handle for long requirements. Furthermore, utilizing a larger dataset would demand significantly more hardware resources, particularly in quantum simulations. Consequently, we implemented various strategies to render the requirements manageable for our experiments. While these data manipulation techniques allowed us to address our

research questions, they influenced with no doubt the outcomes of our experiments.

Firstly, we reduced the number of requirements from 969 to 354 by selecting only on the most frequent non-functional requirements, specifically security (SE), usability (US), operational (O), and performance (PE) requirements. Then, we apply some text normalization techniques. In particular, we simplify the text by eliminating commas, periods, question marks, and other punctuation marks. We remove common English words (e.g., "and", "the", "is") that usually don't add much value or information to the overall meaning of a sentence. We apply the lemmatization technique, reducing words to their base or root. For example, "running" would be lemmatized to "run", and thus reducing the vocabulary size to group similar words together. We calculate the mean size, in terms of words, of requirements in the dataset at this point, and it is a little bit more than 22 words per each requirement. We filter out the requirements counting more than 15 words, resulting in 282 requirements left. Lastly, to handle class imbalance in the dataset we randomly reduce the number of samples in the majority class so that all classes have the same number of samples. By balancing the dataset, the intention is to help prevent bias in the model towards the majority class and improve overall performance in multi-class classification tasks. We end our preprocessing with a total of 216 requirements, 54 per each NFR considered.

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

### 6.3.3 *Research Methodology*

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

Table 6.1: Details of the PROMISE expanded dataset.

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

#### 6.3.3.1 *Study design to answer $RQ_{Nfr}1$*

$RQ_{Nfr}1$ aims to determine the effectiveness of various combinations of word embeddings and shallow ML algorithms for the task of classifying NFRs. The approach is illustrated in Figure 6.3.

To identify the most effective approach, we experiment with multiple embedding techniques followed by a set of ML algorithms. Specifically, the study examines five distinct word embedding techniques, namely TF-IDF, Word2Vec, GloVe, FastText, and BERT, each selected for its unique characteristics and previous applications in similar contexts [67].

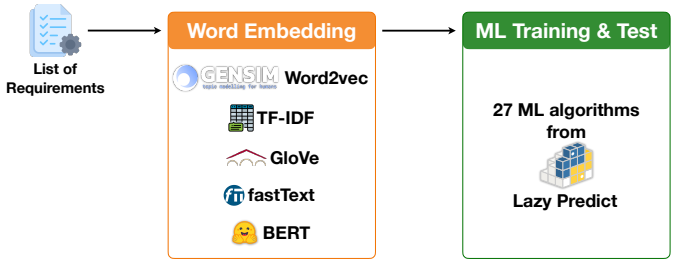


Figure 6.3: Process applied to address $RQ_{Nfr}1$.

TF-IDF [182] is employed for its simplicity and effectiveness in extracting informative features from textual data. It is particularly useful in scenarios where interpretability of word importance is crucial. Word2Vec [146] is included for its ability to capture semantic relationships between words by learning dense vector representations, making it suitable for tasks requiring an understanding of word meaning beyond simple word counts. Specifically, we utilize pre-trained vectors derived from a subset of the Google News dataset with nearly 100 billion words. The model comprises 300-dimensional vectors for 3 million words and phrases, with the latter generated using a data-driven method described in [148]. GloVe [169] is chosen for its capability to model global word co-occurrence statistics, which helps in capturing the semantic structure of the text at a higher level. For this we use pre-trained glove vectors based on Wikipedia 2014 plus Gigaword 5 (6B tokens, uncased). FastText [20, 92, 93]

is exploited for its proficiency in handling sub-word information, making it ideal for processing languages with rich morphological variations. For our experiment, we exploit the 1 million word vectors trained with subword infomation on Wikipedia 2017, UMBC webBase corpus and "statmt.org" news dataset (16B tokens). Lastly, BERT [57], a transformer-based model, is employed for its advanced contextual embeddings, which capture bidirectional context, thus enabling more nuanced understanding of the text, particularly in complex NLP tasks. Context-free models like Word2vec and GloVe produce a single "word embedding" for each word in the vocabulary. As an example, a word like "table" would have the same representation in both "a book on the table" and "a table in the book". In contrast, contextual models create a unique representation for each word, depending on the surrounding words in the sentence. In our work, we only leverage the Bert Tokenizer, in particular the "bert-base-uncased", to embed our requirements.

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

It is worth noting that in our study, we use a collection of requirements coming from different projects [121]. Details about the datasets can be found in Section 6.3.2. This have been chosen because previous studies exploited them [177, 179, 193], although each of the papers exploits the data for different purposes, such as including all NFRs instead of a particular set, and, thus, in different experimental setups.

To assess which is the best combination we perform a hold-out validation, dividing the dataset in train, test and validation set and calculating the metrics listed in Section 6.3.4. So, we firstly take the best model, based on the Accuracy and the F1-score, by using Lazy Predict for each Word Embedding. Then we test it on the hold out set to assess its performance on unseen requirements.

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

#### 6.3.3.2 *Study design to answer $RQ_{Nfr}2$*

The second research question aims to explore the effectiveness of classical models based on tensor networks for the task of classifying NFRs. To address this, we employ string diagrams, which are well-suited for capturing the compositional structure of language in a mathematical way. The approach begins by representing requirements as string diagrams, which are then parameterized into tensor networks. These tensor networks serve as input for classical ML models that are tasked with the classification of NFRs. The entire process involves several steps, each leveraging specific techniques and tools specified in Figure 6.4.

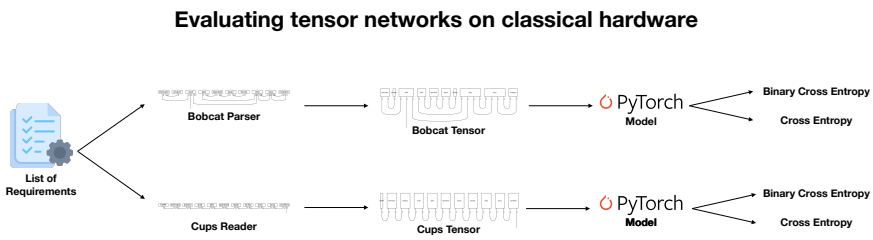


Figure 6.4: Process applied to address $RQ_{Nfr}2$.

Initially, requirements are parsed and represented as string diagrams using a compositional model. In our study, we utilize the DisCoCat (Distributional Compositional Categorical) model, which is known for its ability to combine syntax (from categorical grammars) with semantics (from vector spaces). The string diagrams are created using Bobcat parser, a state-of-the-art statistical CCG parser [45], and Cups Reader. An example output is given in Figure 6.7. The former converts the syntactic structure of the sentence into a pregroup diagram (6.5), the latter combines words linearly using a stair diagram (6.6).

Once the string diagrams are constructed, the next step involves parameterizing these diagrams into tensor networks. This is achieved through the application of a Tensor Ansatz. In this study, we employ a classical Tensor Ansatz, which assigns specific dimensions to the atomic types in the string diagram, effectively converting the abstract diagram into a concrete tensor network. In our experimentation,

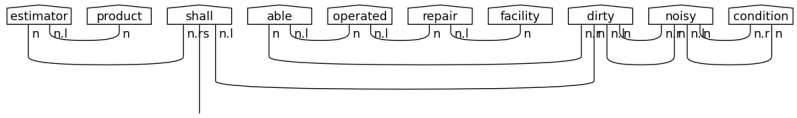


Figure 6.5: Requirement parsed with Bobcat Parser.

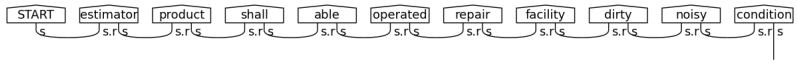


Figure 6.6: Requirement parsed with Cups Reader.

Figure 6.7: Example of requirements converted in string diagram.

we assign "4" as dimension to both the atomic types of "sentence" and "noun". The choice of dimensions is guided by the linguistic properties of the words and the grammatical types they are associated with. The tensor networks derived from the string diagrams are then contracted by the model to obtain the final classification. Tensor contraction is a process that involves summing over shared indices in the network, resulting in a lower-dimensional representation that encapsulates the meaning of the entire sentence.

Examples of tensor networks are given in Figure 6.10, where Figure 6.8 is the parameterized version of the string diagram coming from Bobcat Parser, while 6.9 is the parameterized version of the string diagram coming from Cups Reader. These tensors are then fed into a basic ML model which is trained to classify the requirement as belonging to different categories of non-functional requirements.

The classical ML model is trained on a labeled dataset of non-functional requirements, where each requirement has been pre-categorized. In this experiment, we exploit the PyTorch model provided by lambeq. It is possible of combining tensor networks and neural network architectures to it, however, we use its default definition as provided by the library.

As usually, the training process involves optimizing the model's parameters to minimize a loss function, which measures the difference between the predicted classifications and the actual labels. In particular, we train our model applying an early stop on the best accuracy, adopting "AdamW" as optimizer, with a learning rate of

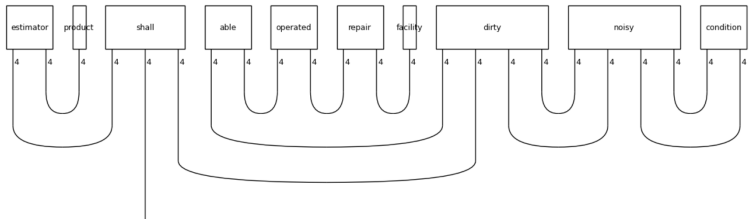


Figure 6.8: Requirement parsed with Bobcat Parser and parameterized with Tensor Ansatz.

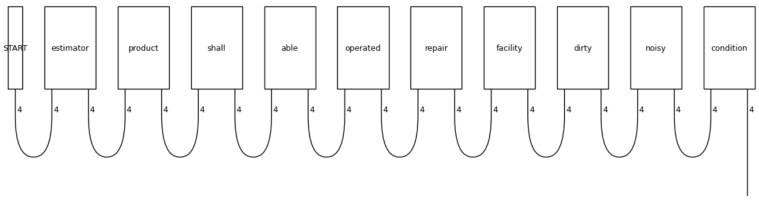


Figure 6.9: Requirement parsed with Cups Reader and parameterized with Tensor Ansatz.

Figure 6.10: Examples of output of Tensor Ansatz from string diagrams.

0.01, 50 epochs and testing two loss functions: Binary Cross-Entropy with Logits Loss and Cross Entropy Loss. After training, the model is evaluated on a separate test set to assess its ability to accurately classify unseen NFRs. The requirements sets are the same used in the experimentation for $RQ_{Nfr}1$, detailed in Section 6.3.2.

#### 6.3.3.3 *Study design to answer $RQ_{Nfr}3$.*

We follow the quantum path in Figure 6.1, performing two kind of quantum simulation: (a) a non-shot based experiment and (b) a noisy shot based experiment (see Figure 6.2), as detailed in Figure 6.11.

The Parsing and Encoding phase is the same adopted in the previous $RQ_{Nfr}2$, ending with the representation of requirements as string diagrams (see Figure 6.7).

Only for (b) the noisy shot based experiment, we rewrite the diagrams to make it simpler for the training of the quantum model simulating scenario. In particular, we adopt the Unify Codomain

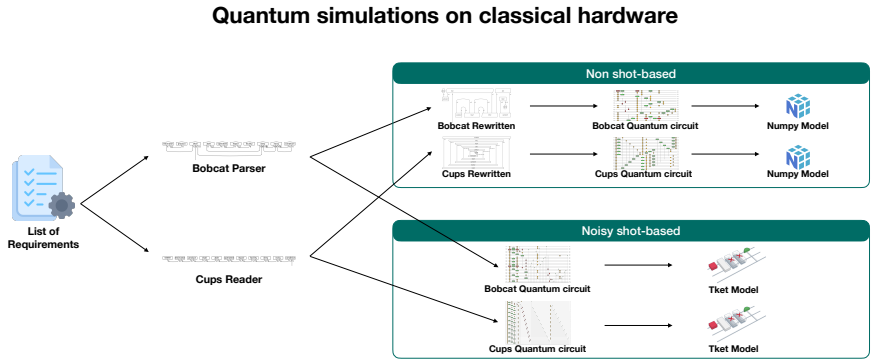


Figure 6.11: Process applied to address $RQ_{Nfr}3$.

Rewriter and the Remove Cups Rewriter. The former rewriter looks at the codomain of the diagram, and if this consists of more than one wires, it merges these wires with an extra box. The latter removes cups from a given diagram, because diagrams with less cups become circuits with less post-selection, which results in faster Quantum ML experiments. An example output of this phase is given in Figure 6.14.

The following step is to parameterize the diagrams into quantum circuits. This is done by applying a quantum circuit ansatz, which maps the abstract structure of the string diagrams to specific quantum gates and qubits. The ansatz determines the qubits required and the type of quantum operations (gates) that will be applied to each component of the string diagram. In this study, we investigate the Instantaneous Quantum Polynomial (IQP) Ansatz, which interleaves layers of Hadamard gates with diagonal unitaries [76].

Note that at this point we have in total four kind of diagrams to deal with. For the experiment (a), i.e. the non-shot based simulation, we have the diagrams output of the Bobcat parser and Cups Reader (Figure 6.5 and Figure 6.6, respectively). For the experiment (b), i.e. the noisy shot based simulation, we have the diagrams output of the rewriting phase (Figure 6.12 and Figure 6.13).

An important choice is made on the setup of the IQP Ansatz, which determines the structures of the created circuits. The setup of the IQP Ansatz is configured through a mapping of atomic types (such as sentences and nouns) to qubits, along with the definition

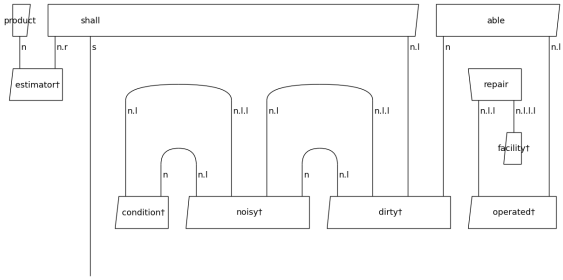


Figure 6.12: Requirement parsed with Bobcat Parser and rewritten with Unify Codomain and Remove Cups rewriters.

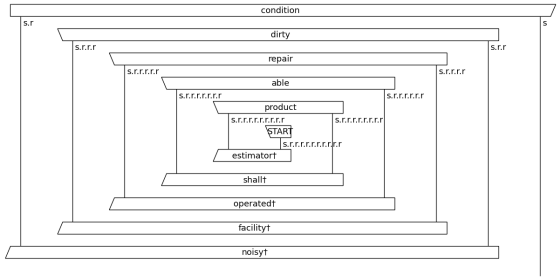


Figure 6.13: Requirement parsed with Cups Reader and and rewritten with Unify Codomain and Remove Cups rewriters.

Figure 6.14: Examples of output of Rewriting phase from string diagrams.

of key parameters that control the structure and complexity of the quantum circuit. In our study, the IQP Ansatz is characterized by a single layer of quantum gates (n_layers=1), the use of multiple single-qubit parameters (n_single_qubit_params=3), and the option to discard additional qubits after processing (discard=True). Firstly, we define a mapping that assigns a specific number of qubits to each atomic type present in the string diagram. In our case “sentence” is mapped to 2 qubits, while “noun” is mapped to 1 qubit. This mapping determines how many qubits are allocated to represent the

quantum states corresponding to sentences and nouns in the diagram. The choice of 2 qubits for sentences allows for a richer representation of their structure, while 1 qubit is used for nouns, reflecting their simpler structure in this context. The "n_layers" parameter is set to 1, indicating that the quantum circuit will consist of a single layer of Hadamard gates interleaved with diagonal unitaries. This setup provides a basic level of entanglement and interaction between the qubits, suitable for capturing relationships in the data while keeping the circuit relatively simple. The "n_single_qubit_params" parameter is set to 3, meaning that each qubit will be associated with three parameterized single-qubit gates (such as rotations). These gates allow for fine-tuning the quantum states of individual qubits, enabling the circuit to capture more detailed information about the input data. This parameterization is crucial for adjusting the quantum circuit to effectively encode the information contained in the string diagrams. Lastly, the "discard=True" option indicates that any qubits that are not needed after the main computation will be discarded. Discarding excess qubits helps reduce the complexity of the circuit and avoids potential issues with post-selection, making the circuit more efficient and easier to execute on quantum hardware.

We use IQP Ansatz configured in this way for all of our experiments. Examples of resulting quantum circuits are given in Figure 6.19. A more detailed picture of each can be found in the shared notebooks in the replication package [7].

The next step is the training and testing. For the (a) non-shot based experiment, we simulate a quantum environment using the Numpy Model, which leverages tensor networks simulations to represent the quantum circuits and their operations. We set "use_jit=True" parameter, enabling Just-In-Time (JIT) compilation, which optimizes the model's performance by speeding up the evaluation of the tensor networks. In this case, we only use the Cross Entropy Loss as the loss function, as it is the most suited one for the multiclass classification task and because we do not want to add another parameter to be analyzed at this stage, in contrast to the previous experimentation for $RQ_{Nfr}2$. The model is trained for 50 epochs, while the SPSA Optimizer [207] is used for optimizing the model's parameters. Note that the optimizer's behavior is controlled by three hyperparameters:

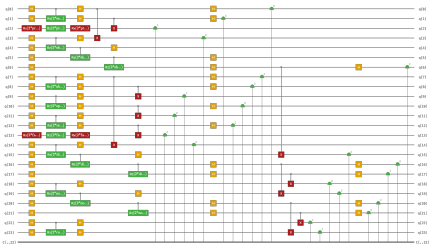

Figure 6.15: Parameterised circuit from diagram in Figure 6.5.

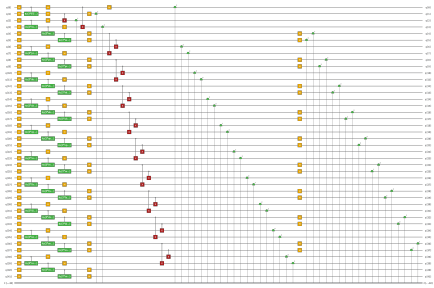

Figure 6.16: Parameterised circuit from diagram in Figure 6.6.

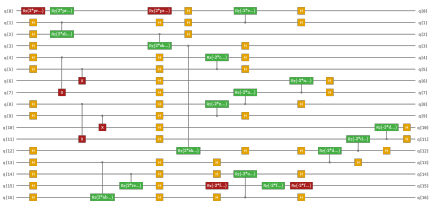

Figure 6.17: Parameterised circuit from diagram in Figure 6.12.

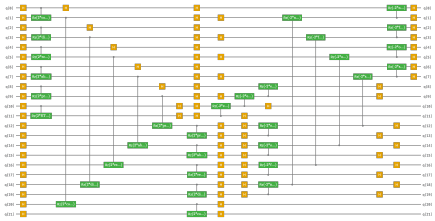

Figure 6.18: Parameterised circuit from diagram in Figure 6.13.

Figure 6.19: Examples of output of IQP Ansatz from string diagrams.

"a", the step size scaling factor or learning rate, set to 0.05; "c", the perturbation scaling factor, set to 0.06; and "A", a stability constant, set to 0.01 * 50 (number of epochs). This is the default configuration suited for most of the cases, and we use this one as our first intention is not to find the most effective setup, but rather to explore the various scenario provided in this field. The training process is carefully monitored, and early stopping on the accuracy metric is employed to prevent overfitting, ensuring the model's robustness and effectiveness in classifying non-functional requirements.

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

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

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

### 6.3.4 *Evaluation Criteria*

To evaluate the models we experiment with, we utilize well-established information retrieval metrics: Accuracy, Precision, Recall, and F1-score [9, 173].

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

We choose these criteria due to their importance in most if not all classification tasks. In our study, Recall should be particularly significant because stakeholders may prioritize a tool that identifies all potential NFRs, even if it results in some incorrect predictions [105]. Moreover, these metrics allow for comparisons between experiments we carry out and with existing or following works in literature, helping to assess and analyze the effectiveness or weaknesses of this approach. The objective is to evaluate the models' accuracy in identifying NFRs and to identify any limitations in this task. For evaluating the results, we consider a model to be effective if it achieves an F1-score greater than 0.7, a threshold that has been used in studies as an indicator of model quality [87, 105].

## 6.4 ANALYSIS OF THE RESULTS

In this section, we present the results of our investigation for each research questions formulated in Section 6.3.1.

### 6.4.1 $RQ_{Nfr}1$. *How effectively can shallow machine learning algorithms classify non-functional requirements when represented with word embeddings techniques?*

We summarize the results in Table 6.2, which provides a comparison of word embedding techniques (TF-IDF, BERT, fastText, Word2vec, and GloVe) and their performance when combined with different ML models. Each technique is evaluated based on the accuracy and F1-score of the models, and a detailed analysis is provided by examining the best model performance on a hold-out test set.

For TF-IDF, performance remains relatively low across all models (Table 6.3). The Random Forest Classifier performs best, achieving an accuracy of 0.39 and an F1-score of 0.36. However, on the hold-out set (Table 6.4), results reveal poor precision and recall, especially for the Security (SE) class, which fails to produce any correct classifications. BERT embeddings show a notable improvement over TF-IDF (Table 6.5). The Ridge Classifier CV delivers the best performance with an accuracy of 0.55 and an F1-score of 0.54.

Despite these gains, hold-out set results (Table 6.6) remain moderate, with Operability (O) and Performance (PE) classes achieving the highest scores. However, challenges persist with the Security (SE) and Usability (US) classes, which record lower F1-scores. Moving to fastText embeddings, we observe superior performance compared to both TF-IDF and BERT. The SVC (Support Vector Classifier) model achieves an accuracy of 0.66 and an F1-score of 0.66. On the hold-out set (Table 6.8), results remain consistent across all classes, with an average F1-score of 0.68. The Security (SE) class demonstrates significant improvement, reaching an F1-score of 0.70, which is higher than what is observed with previous embeddings. Word2vec provides even stronger results, with the Random Forest Classifier achieving an accuracy of 0.77 and an F1-score of 0.78. The hold-out set (Table 6.10) further confirms this, particularly with the Performance (PE) class, which reaches an impressive F1-score of 0.92. The average F1-score across all classes stands at 0.81, highlighting the effectiveness of Word2vec for NFR classification tasks. GloVe embeddings outperform all other techniques, with the NuSVC model delivering the highest accuracy of 0.82 and an F1-score of 0.82. On the hold-out set (Table 6.12), results are outstanding, with an average F1-score of 0.84. Precision and recall remain balanced across all classes, and the Security (SE) class achieves an F1-score of 0.80, making GloVe the most effective embedding technique for the NFRs classification task.

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

Table 6.2: Comparison of selected word embedding techniques with ML models.

Table 6.3: Best models based on TF-IDF.

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

Table 6.4: Test on hold out set of Random Forest Classifier.

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

Table 6.5: Best models based on BERT.

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

Table 6.6: Test on hold out set of Ridge Classifier CV.

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

Table 6.7: Best models based on fastText.

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

Table 6.8: Test on hold out set of SVC.

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

Table 6.9: Best models based on Word2vec.

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

Table 6.10: Test on hold out set of Random Forest Classifier.

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

Table 6.11: Best models based on GloVe.

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

Table 6.12: Test on hold out set of NuSVC.

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

### 6.4.2 *$RQ_{Nfr}2$. How effectively can a basic model classify non-functional requirements when represented as string diagrams and parameterised as tensor networks?*

To answer $RQ_{Nfr}2$ we analyzed the performance of classical Pytorch models using Bobcat-based and Cups-based tensor networks for classifying NFRs. The training process involved two separate setups, one using Binary Cross-Entropy (BCE) loss and the other using Cross Entropy (CE) loss.

Firstly, let's analyze the Bobcat-based experiment. The model's training and validation performance, shown in Figure 6.20 over the course of 43 epochs using BCE loss show substantial improvement in training metrics while validation metrics stabilize. During Epoch 1, the training accuracy was 0.23, with an F1-score of 0.52, and by Epoch 10, training accuracy had increased to 0.97, and the F1-score had reached 0.96. However, the validation accuracy remains at around 0.27, and the validation F1-scores hovered between 0.25 and 0.36 during the entire training period. The early stopping criterion is triggered at Epoch 43, selecting the best model at epoch 20 and suggesting that further training would likely result in overfitting.

After training, we test the model on a hold-out dataset, obtaining the results shown in Table 6.14. A class-wise performance revealed that class (O) achieved the best precision and recall, with an F1-score of 0.35.

Using Cross Entropy (CE) loss, the training results (Figure 6.21) show a similar trend to the BCE training setup, with better training metrics but a slower convergence for validation metrics. During Epoch 1, the training accuracy started at 0.23, and by Epoch 10, it increased to 0.98. The validation accuracy stabilize around 0.36, with an F1-score in the 0.34 to 0.36 range by Epoch 10. Early stopping was triggered at Epoch 24, with the model reaching its peak performance at Epoch 8. We test the best model on the hold-out set, producing the results shown in Table 6.15. Although the overall accuracy is low, the CE-based model shows a slight improvement in class 0 predictions, but the other classes, particularly usability class, still shows weak precision and recall.

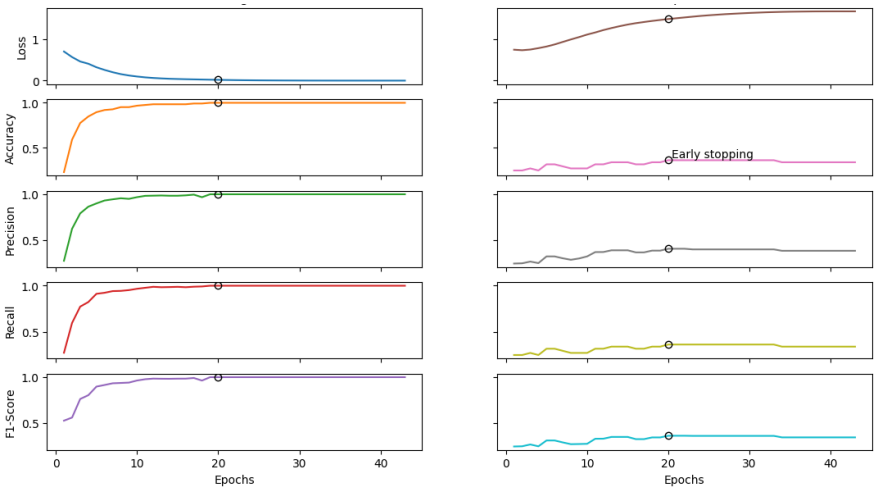


Figure 6.20: Performance in training with BCE on training set (left) and validation set (right).

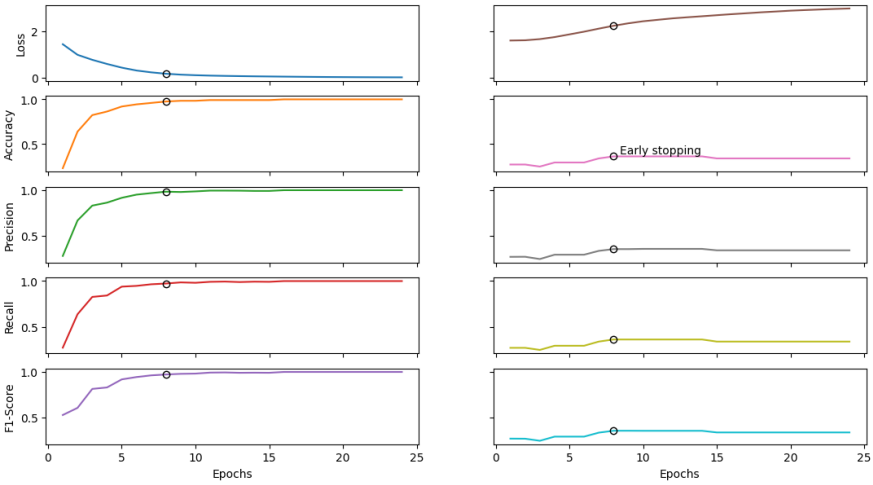


Figure 6.21: Performance in training with CE on training set (left) and validation set (right).

Figure 6.22: Performance during training with Bobcat-based tensor networks.

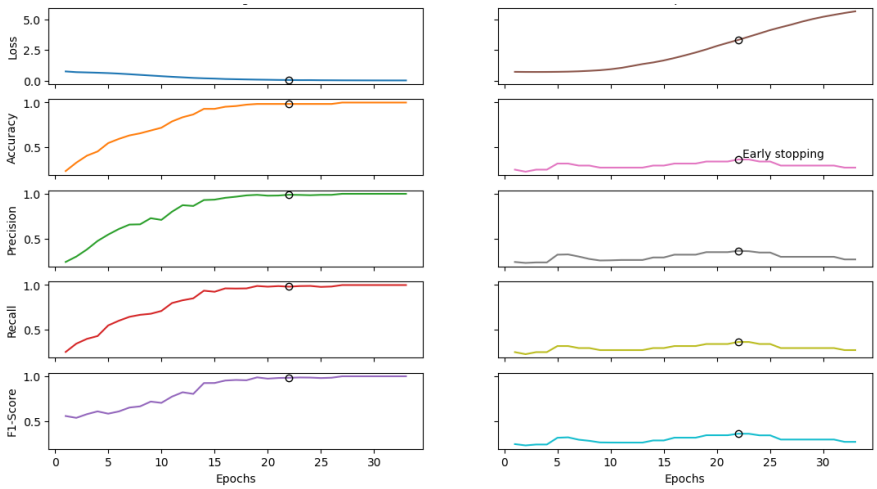


Figure 6.23: Performance in training with BCE on training set (left) and validation set (right).

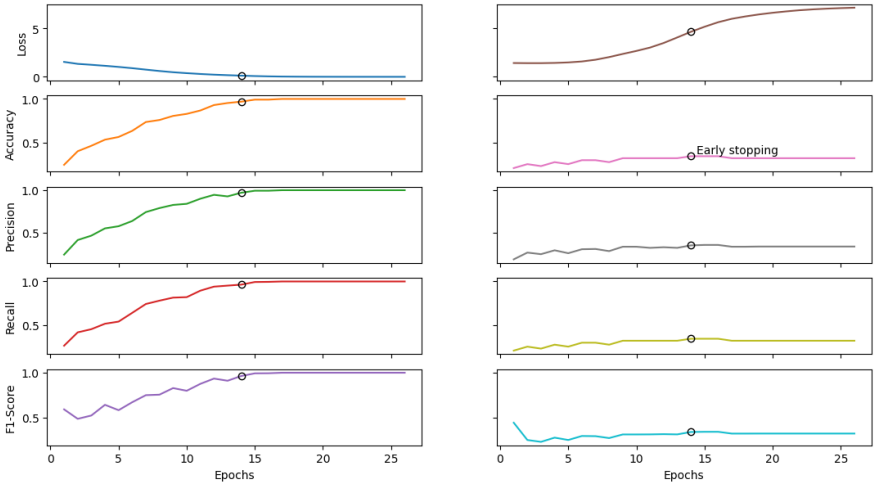


Figure 6.24: Performance in training with CE on training set (left) and validation set (right).

Figure 6.25: Performance during training with Cups-based tensor networks.

Table 6.13: Comparison of Pytorch models trained on Bobcat-based tensor networks.

Table 6.14: Results using BCE with Logits Loss during training and test on hold out set.

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

Table 6.15: Results using CE Loss during training and test on hold out set.

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

During training on Cups-based tensor networks with the BCE loss, the model shows notable improvements across the first few epochs but begins to overfit as validation metrics stabilize, as shown in Figure 6.23. At Epoch 1, results are modest, with a training Accuracy of 0.23 and an F1-score of 0.56. The validation accuracy is 0.25, with a lower F1-score of 0.24. At Epoch 22 the best model is identified, with training accuracy peaking at 0.98 and an F1-score of 0.98. Validation accuracy is 0.36, with an F1-score of 0.36, signaling limited generalization capacity. Early stopping is triggered, and this model is saved for evaluation on the hold-out set. This model is evaluated on the hold-out set, yielding the results displayed in Table 6.17. Again, the model achieves the best performance in operability class, while on other classes it shows relatively lower results, by reaching an average F1-score of 0.23, indicating moderate performance across the classes.

For the experiments using CE loss, the Pytorch model demonstrates modest initial training performance but faces challenges in achieving generalization, as seen in the validation performance in Figure 6.24. At Epoch 1 training accuracy is 0.24 with an F1-score of 0.59, while the validation accuracy is 0.20 and the F1-score is 0.44. At Epoch 14 the model is saved as the best-performing model, with 0.97 training accuracy and an F1-score of 0.96. Validation accuracy is 0.34, and the F1-score is 0.34. Early stopping is triggered at this point. The best CE model is evaluated on the hold-out set, providing the results shown in Table 6.18. Also in this case, on the operability class, the model has the highest F1-score at 0.56, followed by security class with 0.50.

Table 6.16: Comparison of Pytorch models trained on Cups-based tensor networks.

Table 6.17: Results using Binary Cross Entropy with Logits Loss during training and test on hold out set.

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

Table 6.18: Results using Cross Entropy Loss during training and test on hold out set.

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

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

### 6.4.3 $RQ_{Nfr}3$. *How effectively can quantum models classify non-functional requirements when represented as string diagrams and parameterised as quantum circuits?*

As anticipated in Section 6.3.1, to answer $RQ_{Nfr}3$ we perform two experiments: (a) non-shot based experiment and (b) a noisy shot based experiment (see Figure 6.2). In the experiment (a), the results we get during train are shown in Figure 6.28.

In particular, the results of training a Numpy model based on quantum circuits generated from the Bobcat parser show moderate performance with consistent challenges across the epochs (Figure 6.26). In the initial epoch, the model starts achieving a training accuracy of 0.29 and a validation accuracy of 0.23.

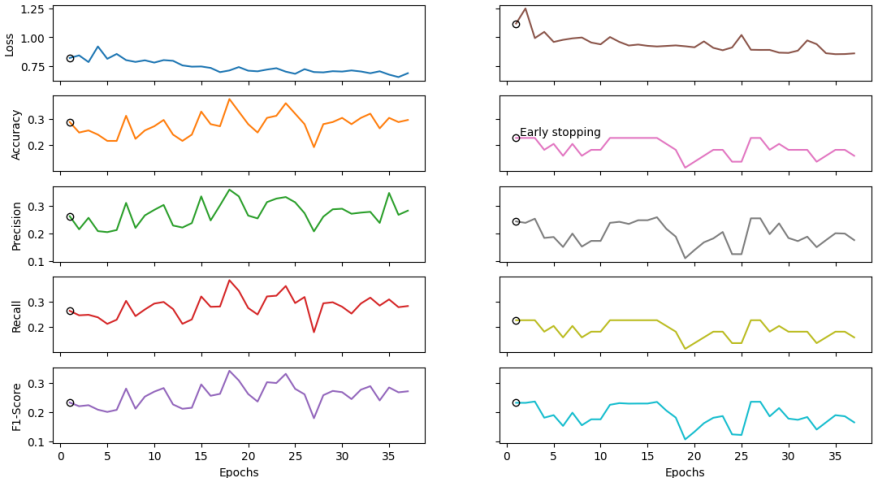


Figure 6.26: Performance with Bobcat-based quantum circuits on training set (left) and validation set (right).

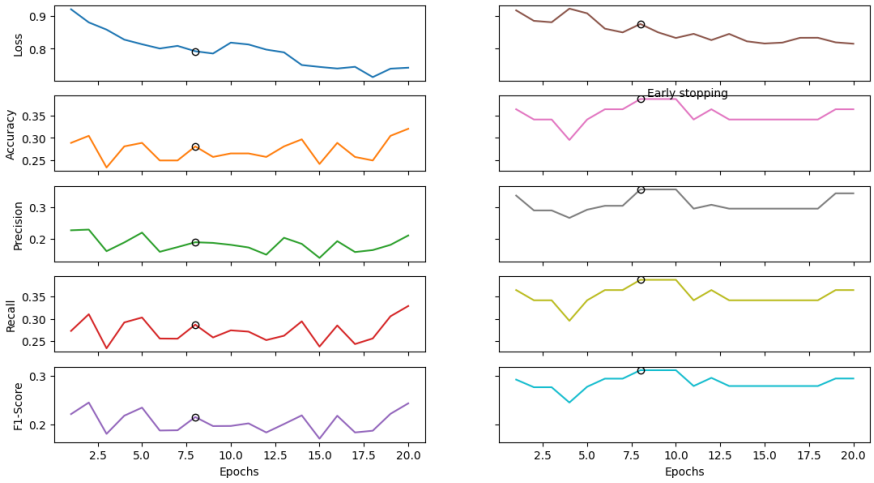


Figure 6.27: Performance with Cups-based quantum circuits on training set (left) and validation set (right).

Figure 6.28: Performance during training with Numpy model on quantum circuits.

The corresponding F1-scores of 0.23 for training and 0.23 for validation indicate an underfitting trend. Over the next few epochs, the model's performance fluctuates slightly, with training accuracy peaking at around 0.31 in epoch 7, while validation accuracy remains stagnant at 0.23. The precision and recall metrics similarly reflect the model's struggle to generalize. For instance, the precision for validation oscillates around the 0.24-0.26 range across most epochs, while recall stays relatively stable at 0.23, with minor improvements. The model continues to suffer from low F1-scores, indicating it struggles with both overfitting and generalization issues throughout training.

The best model, selected after the first epoch, is applied to the hold-out set, with the results shown in Table 6.20. The evaluation results indicate a hold-out accuracy of 0.30, which is marginally better than the validation accuracy but still quite low. The precision (0.33) and recall (0.30) suggest that while the model can make some correct predictions, it perform consistently across all classes. Indeed, the F1-scores between classes fluctuate in range 0.24-0.35, indicating that some features help the model to distinguish between classes.

The training results from the Cups-based quantum circuits experiment (Figure 6.27) exhibit fluctuating performance across the epochs, with gradual improvement in validation metrics but still limited classification effectiveness. Epoch 1 starts with a training accuracy of 0.29 and validation accuracy of 0.36 reflect the model's initial struggle to classify the non-functional requirements effectively. The F1-scores are 0.22 (training) and 0.29 (validation), indicating poor generalization. Over next epochs, the training precision and recall fluctuate, showing modest increases, especially by Epoch 8, where validation accuracy reaches 0.39, with an F1-score of 0.31. However, the training precision remains low throughout, and the model shows difficulties in learning patterns effectively from the dataset. Once the best model (saved at Epoch 8) is applied to the hold-out set, the overall performance remains modest (Table 6.21). Operability class performs the best in terms of both recall and F1-score, though precision remains moderate. In Performance class, the model struggles with detecting this, resulting in a low recall and F1-score, while recall is higher in Security class than Performance class.

Table 6.19: Comparison of Numpy models trained on quantum circuits.

Table 6.20: Results using Numpy models with train on Bobcat-based quantum circuits and test on hold out set.

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

Table 6.21: Results using using Numpy models with train on Cups-based quantum circuits and test on hold out set.

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

In the (b) shot-based experiment, we train the Tket quantum model using quantum circuits created from both the Bobcat parser and Cups Reader and executed on the AER Backend. The results of Tket training on Bobcat-based quantum circuits demonstrate variability in model performance across different epochs, with generally low precision, recall, and F1-scores, indicating limited learning from the data (Figure 6.29). Epoch 1 starts with a low training accuracy of 0.25 and validation accuracy of 0.27. The initial F1-scores are 0.17 (training) and 0.19 (validation), showing that the model is not learning effectively. As the epochs progress, the training precision and recall fluctuate without significant improvement. Epoch 11 shows the best performance with training accuracy of 0.30 and validation accuracy of 0.32, alongside an F1-score of 0.24 for the validation set. The model stabilizes by this epoch, but performance remains limited. After applying the best model (from Epoch 11) to the hold-out set, the overall performance remains weak, as can be seen in Table 6.23. The model performs best on Operability class, with relatively high recall but low precision. On the Performance class the results are moderate, with low recall limiting the model's effectiveness. The remaining classes both exhibit 0.00 on precision, recall, and F1-scores, indicating the model's complete inability to classify these classes.

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

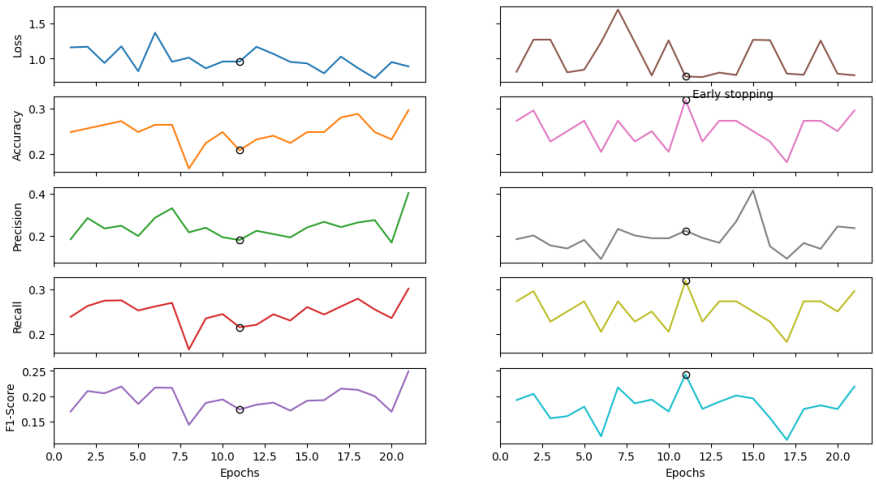


Figure 6.29: Performance with Bobcat-based quantum circuits on training set (left) and validation set (right).

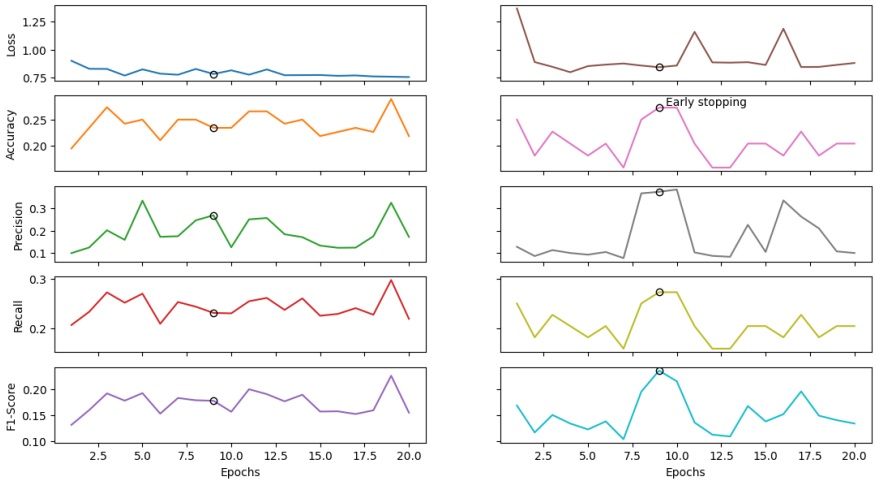


Figure 6.30: Performance with Cups-based quantum circuits on training set (left) and validation set (right).

Figure 6.31: Performance during training with Tket model on quantum circuits.

that the model struggles to classify non-functional requirements effectively (Figure 6.30). Epoch 1 starts with poor performance in both precision and recall. Training accuracy is 0.20, while validation accuracy is 0.25, showing that the model is struggling to generalize from the data. The F1-score for training is 0.13 and 0.17 for validation. As training continues, precision and recall remain inconsistent. The best training performance is seen in Epoch 9, with training precision reaching 0.27 and F1-scores of 0.24 on the validation set, reflecting moderate improvement in some classes. Early stopping is triggered after Epoch 20, as the model fails to make significant gains in performance, and the best model from Epoch 9 is saved. After applying the best model from Epoch 9 to the hold-out set, the performance varies between classes, with the Operability class achieving 0.32 precision and 0.64 recall, but Performance and Usability class exhibit 0.00 precision, recall, and F1-scores, showing that the model fails to detect certain classes.

Table 6.22: Comparison of Tket models trained on quantum circuits.

Table 6.23: Results using Tket model with train on Bobcat-based quantum circuits and test on hold out set.

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

Table 6.24: Results using Tket model with train on Cups-based quantum circuits and test on hold out set.

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

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

## 6.5 DISCUSSION AND RESEARCH ROADMAP

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

### 6.5.1 *Lesson learned*

The results of $RQ_{Nfr}1$ inform us that the combination of shallow ML algorithms with word embeddings result in very high performing ML models, classifying unseen non-functional requirements during the training phase with precision, recall, and f1-score metrics exceeding 0.80. In particular, GloVe and Word2Vec are predominant, with the former performing slightly better, although statistical tests should be conducted to confirm the significant difference. However, our focus is not to understand which combination is the most effective for the task at hand, but to understand where the new solutions offered by both

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

👉 In summary, $RQ_{Nfr}1$ results differentiate statistical embeddings, such as GloVE, Word2vec, FastText, which consider words independently of context, from embeddings based on word frequency, such as TF-IDF, and from contextual embeddings, such as BERT. In particular, statistical embeddings achieve more than satisfactory results, while the rest have difficulties, due to both the volume of the data under consideration and the preprocessing steps used during the study.

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

Experimentation results exploited to answer $RQ_{Nfr}2$ suggest that the representation via Cups Reader performs better than Bobcat Parser, especially using Cross Entropy as loss function in PyTorch model training (Table 6.18). Note that for all our experiments we used the default version of the model, which simply exploits the tensors contraction to provide the output. This implies several considerations. First, we can evaluate the results obtained from simply contracting

the tensors obtained from Bobcat Parser and Cups Reader, allowing a "direct" comparison between the different representations in string diagrams of the requirements. Next, since the basic model is a simple contraction operation, we can consider this model not very different from shallow ML models that are still based on more or less complex mathematical functions, allowing us to compare with the results obtained in response to $RQ_{Nfr}1$. Finally, since the model is based on PyTorch, it is possible to build from it more complex neural networks that could positively or negatively affect the final performance, in addition to the possibility to load the weights and symbols from a training checkpoint, allowing for an incremental learning implementation.

As a matter of fact, in the experimentation to answer $RQ_{Nfr}1$ we evaluated the efficiency of different ML algorithms, while for $RQ_{Nfr}2$, exploiting the basic model, we did not obtain astonishing performance. However, the results obtained from Cups Reader (Table 6.18) are better in 3 classes (Operability, Security, and Usability) out of 4 than the combination of BERT and shallow ML (Table 6.6), which presents better results only in identifying the Performance class.

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

The representation of requirements in string diagrams opens the way for comparison with the results obtained in the experiment carried out to answer $RQ_{Nfr}3$, in which we represent string diagrams in quantum circuits. Clearly, here the factors influencing the requirements classification are even more numerous than in the previous case. In fact, the parameterization of the tensor diagrams in the experimentation performed to answer $RQ_{Nfr}2$ was done through the definition of a sentence and noun mapping within the requirements by assigning dimension 4 to each, since 4 are the possible classes to which each belongs. However, this is also a questionable design

choice, as we could have defined larger or smaller dimensions and modified the PyTorch model accordingly to shape the output of the neural network in 4 possible classes.

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

Considering these factors, additional choices are made in the simulation of quantum environments on classical hardware. First of all, the Numpy model we use in the non-shot based simulation converts quantum circuits into tensor networks, which on the one hand allows a comparison with the experimentation done in $RQ_{Nfr}2$ since both are based on tensor network, and on the other hand is a bit counter-intuitive if one thinks of wanting to exploit the potential of quantum computation for the task of requirements classification.

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

Analyzing the results for $RQ_{Nfr}3$, obtained using the same setup adopted for $RQ_{Nfr}1$ and $RQ_{Nfr}2$ (i.e. same dataset, same task, same metrics, and same hold out validation), we can conclude that in both types of quantum environment simulation the results leave something to be desired. However, despite the probabilistic nature of these simulations, in some cases the results are shown to be comparable to those obtained in the other experiments. In detail, only in the non-shot based simulation we have obtained a model capable of recognizing all types of NFRs by exploiting Bobcat-based representation (Table 6.20), albeit with relatively poor performance. In the remaining cases, there are classes that fail to be identified. In the shot-based simulation, the Operability class exhibits high recall in both cases, even better than the non-shot based simulation, but the remaining results on the other classes are really scarce.

When comparing the best results obtained for $RQ_{Nfr}2$ (Table 6.18) with the best results for $RQ_{Nfr}3$ (Table 6.20), the latter perform relatively worse, with a deviation of 0.10 on F1-score. However, although they are not the best, they are still better than the results obtained using TF-IDF and ML techniques in combination.

👉 In the non-shot based simulation, Numpy model converts quantum circuits into tensor networks, aligning it with $RQ_{Nfr}2$'s approach, which also uses tensor networks. However, this may seem counter-intuitive given that quantum computation's potential is not fully leveraged. In shot-based simulations, circuits configuration, compilation passes, and the number of shots play critical roles in influencing outcomes. While the results for $RQ_{Nfr}3$ do not outperform those of $RQ_{Nfr}2$, they show comparability, especially in identifying specific NFR classes. However, overall performance is lower than the tensor-based models in $RQ_{Nfr}2$, with a 0.10 F1-score deviation, though still better than the combination of TF-IDF and ML methods.

### 6.5.2 *Future Research Directions*

Based on the methodology we adopted, in the following we want to share how researchers can organize the future directions for research in quantum-based non-functional requirements (NFR) classification, which is graphically summarized in Figure 6.32.

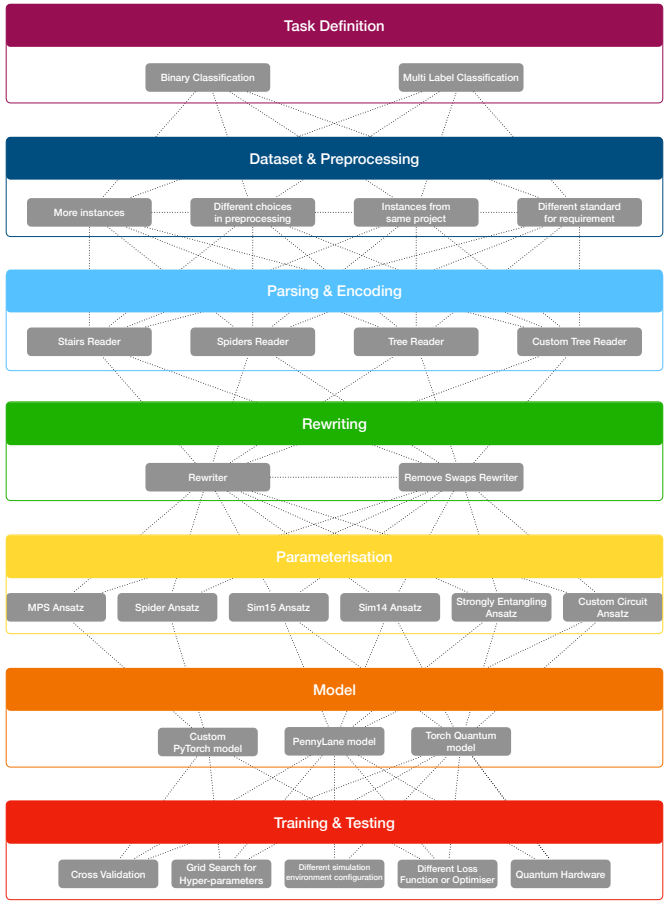


Figure 6.32: Roadmap for unexplored application of Quantum NLP for the NFRs classification task.

Firstly, future work could expand on task complexity by experimenting with binary or multi-label classification rather than the multi-class classification approach used in this work. This would

enable a more nuanced understanding of the interdependencies between NFR categories, which often overlap in real-world applications.

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

In future work, a wider variety of parsers and encoders could be explored. This might include leveraging advanced readers such as the Stairs Reader, Spiders Reader, or Tree Reader, and even custom parsers for requirement extraction and encoding. Such advancements could improve the accuracy of initial representations, a crucial factor when inputting data into quantum circuits or when parameterising such tensor networks. Exploring new quantum or classical encoding strategies may help align the peculiarities of requirements data with quantum computing's strengths.

In the rewriting phase, new strategies could be explored to optimize the string diagrams. While some Rewriters helped us to simplify string diagrams, future work could investigate other rewriting techniques or even custom rewriters that better optimize them to obtain simplified quantum circuits for specific backends. This would minimize noise and errors during quantum operations, leading to more accurate results. More sophisticated quantum rewriting algorithms can also help circuits to execute more efficiently on simulators or actual quantum hardware.

One area of research that offers significant potential for improvement is circuit parameterization. Beyond the other available ansätze

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

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

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

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

## 6.6 THREATS TO VALIDITY

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

**Construct Validity**. Construct validity pertains to how well the evaluation methods measure the concepts they are intended to assess, ensuring that the observations and inferences drawn from the study are appropriate [28]. In this work, we employed metrics from the Scikit-learn library, including the `accuracy_score`[5] for Accuracy, `precision_score`[6] for Precision, `recall_score`[7] for Recall, and `f1_score`[8] for F1-score.

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

**Internal Validity**. Internal validity is concerned with the degree to which the results can be attributed to the interventions applied, rather than other factors [28]. In our study, we assume comparability among the models because they are derived from the same libraries, such as Scikit-learn in $RQ_{Nfr}1$, and lambeq in $RQ_{Nfr}2$ and $RQ_{Nfr}3$. Since causality could pose a threat to internal validity, we are aware that the results are the product of chance, mainly because the choices made in the course of the experiments are varied, starting from the preprocessing of the dataset to the selection of models to be adopted in the different scenarios. Further experiments are needed, and to support that we design a possible road map in Section 4.5 to help and stimulate the research in the field.

**External Validity**. External validity relates to the generalizability and reproducibility of the study's findings [28], which determines the applicability of our results in real-world scenarios.

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

Our approach utilizes Python-based models, specifically those implemented using available Python libraries. The methods employed in this study may not be directly transferable to other programming languages, and since we do not apply validation techniques focused on generalizability like cross validation, the external validity may be dependent on the specific training data. However, the datasets used span over various projects within software specifications, supporting the applicability of the approach in different industrial settings. To facilitate the replication of our study, all tools, scripts, and data have been made publicly available [7].

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

**Conclusion Validity**. Conclusion validity refers to the soundness of the inferences drawn from the data and the reliability of the conclusions based on the employed metrics [28].

We recognize that the conclusions regarding the effectiveness of the ML models in our RQs are based on metrics like Precision, Recall and so on. We acknowledge that these metrics alone may not capture the full performance potential of the models. Additional experimentation and statistical tests could provide a deeper insight into the model's efficacy, thereby refining the conclusions presented in this study.

# PART III

# PARADISO

*"La gloria di colui che tutto move*
*per l'universo penetra, e risplende,*
*in una parte più e meno altrove.*

*Nel ciel che più de la sua luce prende*
*fu' io, e vidi cose che ridire*
*né sa né può chi di là sù discende;*

*perché appressando sé al suo disire,*
*nostro intelletto si profonda tanto,*
*che dietro la memoria non può ire.*

*Veramente quant'io del regno santo*
*ne la mia mente potei far tesoro,*
*sarà ora materia del mio canto."*

*Dante Alighieri, Paradiso, Canto I, vv. 1 - 12.*

# 7 DISCUSSION AND RESEARCH DIRECTIONS

This chapter first answers the three high-level research questions ($\mathbf{RQ}_A$– $\mathbf{RQ}_C$ ) presented in Chapter 1. Then, it further elaborates on the findings observed in the studies presented in Chapters 3, 4, 5, and 6, reflecting on the new research gaps that originated from them and calling for contributions and novel research artifacts to improve the detection of NFRs. This chapter discusses different aspects, grouped into three thematic areas. Each area calls for (1) further empirical investigation on which aspects of language structure improve the performance of the different solutions, (2) analysis of the features that allow hardware resources to face data scarcity, and (3) the introduction of novel automated techniques to deal with the contextual information hidden in requirements.

## 7.1 ANSWERS TO RESEARCH QUESTIONS

**RQ**$_A$ To what extent can analyzing the structure of requirement language enhance the detection of NFRs?

Chapters 3 [J1] and 6 [J4] analyzed several key characteristics concerning the language structure, looking at how the requirements are syntactically and semantically written. In particular, the works focused on the relevance of considering entities, parts of speech and dependencies as features for ML and DL solutions in detecting privacy concerns (Chapter 3) [J1] and the impact that different requirement representations may have in inform ML algorithm and neural networks (Chapter 6) [J4]. Indeed, if in the former case (Chapter 3) such features lead to optimal performance and better solution when compared to other classical approaches, in the latter case (Chapter 6) we observe that the way requirements are represented significantly impact on the accuracy of the models.

The results obtained inform us that such novel representations, i.e string diagrams, may be better than (1) statistical lexical frequencies techniques like TF-IDF and (2) contextual embedding like BERT, but worse than pre-trained vector solutions like Word2Vec, GloVe and FastText. However, this is somehow expected because pre-trained vector extract knowledge from larger corpora of text, instead string diagrams are based just on texts composing requirements, as in the case of TF-IDF.

👉 **Answer to RQ$_A$.** Analyzing the structure of requirement language significantly enhances the detection of NFRs by leveraging semantic and syntactic patterns. Indeed, DL methods, like CNNs, outperform traditional ML by modeling requirement based on their structures and semantic. However, the choice of features is critical; considering solely the lexical characteristics led to worst performance. Furthermore, while tensor networks and quantum methods show potential for deeper structural modeling, their limited effectiveness when compared to advanced solution based on word embedder highlights the need for further optimization and refinement, even though the representation in string diagrams is comparable with contextual embeddings techniques and performs better than techniques that consider words frequencies.

🔍 **RQ$_B$** How much can the availability of hardware resources counterbalance the data scarcity?

Chapters 3 [J1], 5 [J3] and 6 [J4] explore how the hardware resources improves the NFRs detection in comparison to traditional ML solutions when dealing with data scarcity. In particular, Chapters 3 [J1] highlight how resources such as GPUs and TPUs facilitate the training of DL models by enabling more complex architectures (e.g., CNNs and Transfer Learning models) to process limited datasets effectively. Then, the work on security in Chapter 5 [J3] highlights that while hardware plays a critical role in enabling pre-trained transformer models to be fine-tuned on small datasets, the quality of pre-training and the alignment of the pre-trained model's knowledge with the target domain are equally vital.

Lastly, experiments in Chapter 6 [J4] using tensor networks and quantum models, which inherently require significant computational power, reveal that while hardware enables the exploration of advanced architectures, the models are still limited by the small dataset size, suggesting that even powerful computational approaches cannot fully overcome the challenges posed by unrepresentative data.

👉 **Answer to RQ$_B$.** The availability of robust hardware resources significantly mitigates the challenges posed by data scarcity, but its impact is closely tied to the methods employed and the availability of high-quality pre-trained models. Advanced hardware enables the use of computationally intensive techniques, such as Transfer Learning (e.g., $PD_{TL}$ for privacy detection) and pre-trained transformer models like BERT (as demonstrated in the security detection work), which are particularly adept at extracting meaningful patterns from small datasets. However, while hardware resources amplify the effectiveness of such advanced methods, they cannot fully compensate for the absence of relevant amount of data, as emphasized from the insights coming tensor networks and quantum methods, as these methods struggled to generalize despite being resource-intensive.

🔍 **RQ$_C$** To what extent can we introduce domain agnostic NLP-based technologies and techniques?

Chapters 3 [J1], 4 [C3],5 [J3] and 6 [J4] provide an attempt of the applicability of domain-agnostic NLP-based technologies and techniques. In particular, Chapter 3 [J1] shows that by leveraging pre-trained models, such as convolutional neural networks trained on generalized text data, $PD_{TL}$ was able to achieve high performance independently from the domain application of requirements. Chapter 4 emphasizes the critical role of domain detection in building a recommender systems that address fairness concerns. While the methods employed, such as NLP pipelines, were domain-agnostic to some extent, the need for detecting domain-specific biases underlines that domain awareness is essential for the meaningful application of domain-agnostic methods.

Chapter 5 highlights the potential of pre-trained, domain-agnostic transformer models for achieving high performance across domains. BERT, trained on massive amounts of general-purpose text (i.e. CVE and CWE descriptions, as well insecure code examples), exhibited strong results in identifying security-related content. Lastly, Chapter 6 demonstrates that word embeddings like GloVe, Word2Vec, and FastText significantly enhance model performance, even in data-scarce settings. These embeddings, trained on large general-purpose corpora, provided meaningful representations of words, enabling improved classification of NFRs such as Operability and Usability in multiclass classification context.

👉 **Answer to RQ$_C$.** Domain-agnostic NLP-based technologies and techniques can be effectively introduced across diverse domains to detect and classify NFRs, but their success depends on a balanced integration of general-purpose models with domain-specific adaptations. Pre-trained models like BERT, word embeddings like GloVe, Word2Vec and FastText, and techniques like transfer learning demonstrate significant promise in generalizing across domains, achieving high performance even in data-scarce settings, even though adaptation remain critical for optimal performance in specialized tasks.

## 7.2 FINDINGS OF THE RESEARCH

Based on the answers to the research questions, the key findings of the thesis can be articulated as follows.

**Semantic and Syntactic Analysis Enhances NFRs Detection.** Deep Learning outperforms traditional ML. CNN-based methods demonstrated their strength in capturing semantic and syntactic nuances within requirement language, outperforming traditional ML approaches. These models excel in understanding contextual relationships between words, which is essential for identifying NFRs, such as privacy requirements, embedded within user stories (e.g., $CNN_{NLP}$ achieved better accuracy than traditional models like Logistic Regression or Random Forest).

Here, feature selection is crucial: the results emphasize that the choice of features is a deciding factor for detection accuracy. Lexical features, such as privacy words, although intuitive, limit the model's performance when used in isolation.

This limitation arises from the inability to capture contextual and structural relationships, which are pivotal for detection of NFRs.

On the other hand, emerging methods show mixed results: tensor networks and quantum methods, designed to explore deeper linguistic structures through mathematical abstractions (e.g., Bobcat Parser, Cups Reader), exhibited only moderate success. These methods struggled with generalization, especially when compared to contextual embeddings like GloVe or Word2Vec. The findings indicate that while these advanced methods have potential, they require optimization to match the effectiveness of established NLP approaches.

🔑 *NFR detection benefits immensely from leveraging both semantic and syntactic structures, and solutions that combine contextual embeddings and DL techniques provide the best results. However, advanced methods like tensor networks and quantum approaches need substantial refinement before they can be widely adopted.*

**Hardware Resources Amplify the Impact of Advanced Methods.** Advanced hardware resources, such as GPUs and TPUs, play a vital role in supporting computationally intensive approaches like CNNs and transformers, facilitating their implementation and usage. For example, $PD_{TL}$ in privacy detection leveraged Transfer Learning to achieve high accuracy even with small datasets by relying on pre-trained CNN architectures.

In security detection, BERT — a domain-agnostic transformer model — was fine-tuned on a small security-specific dataset. Hardware resources allowed this fine-tuning process to be computationally feasible, bridging the gap between general-purpose pre-trained models and domain-specific requirements.

This computational cost limits but not delete the issue of data scarcity: while robust hardware enables the exploration of advanced architectures, it cannot fully overcome the challenges of limited or unrepresentative data. For example: tensor networks and quantum methods, despite being resource-intensive, underperformed due to

their inability to generalize with small datasets. The success of Transfer Learning and transformers like BERT underscores the importance of pre-trained models that already encode meaningful patterns.

🔑 *Hardware resources are a powerful enabler for addressing data scarcity, but their effectiveness is contingent on the availability of high-quality pre-trained models and the inherent capabilities of the employed methods. Computational power alone cannot replace the need for representative datasets.*

**Domain-Agnostic NLP Techniques Require Adaptation.** Domain-agnostic pre-trained models, such as BERT and Transfer Learning strategies, demonstrated strong results in NFR detection across diverse domains. For instance, $PD_{TL}$ for privacy detection effectively leveraged generalized CNN architectures, achieving domain-agnostic performance, while contextual embeddings like GloVe, Word2Vec, and FastText significantly improved NFR classification, offering robust representations of words even in unseen contexts.

While domain-agnostic models are effective, their performance can be significantly enhanced with domain-specific adaptations. For example, fine-tuning BERT on security-specific datasets enabled it to excel in identifying security-related requirements. Fairness detection required domain awareness to uncover domain-specific biases, despite employing largely domain-agnostic NLP pipelines.

Domain-agnostic techniques provide scalability and adaptability across multiple domains but are often complemented by domain-specific enhancements for optimal results. The interplay between these two approaches enables the efficient application of NLP technologies in specialized tasks.

🔑 *Domain-agnostic NLP techniques form a scalable and versatile foundation for NFR detection, but incorporating domain-specific adaptations is critical for ensuring high performance.*

**The Potential and Limitations of Transfer Learning.** Transfer Learning emerged as a standout strategy for addressing data scarcity. By leveraging pre-trained models designed for related tasks, Transfer Learning enabled the reuse of existing knowledge to build effective models in new contexts. For instance, $PD_{TL}$ achieved superior performance in privacy detection by combining pre-trained CNN models with task-specific adaptations. Transfer Learning facilitates efficient

model development in domains with limited labeled data, reducing the need for extensive, domain-specific datasets.

Its success in this thesis highlights its potential for scalability across various NFR detection tasks. However, it is closely tied to the quality and relevance of the pre-trained models used. Misaligned pre-training or insufficient task-specific fine-tuning can hinder its effectiveness, as seen in some domain-specific applications requiring extensive customization.

🔑 *Transfer Learning is a transformative approach for NFR detection, particularly in data-scarce settings. However, its success hinges on the availability of high-quality pre-trained models and careful task-specific tuning.*

**Balancing Generalization and Specialization in NLP Applications.** General-purpose models, such as contextual embeddings and domain-agnostic pipelines, demonstrated significant scalability and adaptability. These approaches provide a strong foundation for tackling diverse NFR detection challenges without the need for extensive domain-specific data.

Despite the effectiveness of general-purpose models, specialized tasks often require domain-specific adaptations. For example, fairness detection relied on understanding domain-specific biases to build effective recommender systems. Furthermore, security detection benefited from fine-tuning general-purpose transformers like BERT on domain-specific datasets.

The findings suggest that the most effective strategies combine domain-agnostic techniques with domain-specific adaptations. This balance ensures scalability while addressing the unique characteristics of specialized tasks.

Note that previous findings is primarily concerned with the applicability and scalability of domain-agnostic methods across different domains. This, instead, focuses on the trade-offs and challenges of balancing general-purpose models (generalization) with task-specific adaptations (specialization) to achieve optimal results.

🔑 *The interplay between generalization and specialization is critical for maximizing the potential of NLP-based technologies in NFR detection. Combining the strengths of domain-agnostic models with task-specific fine-tuning can enable robust and scalable solutions.*

## 7.3 IMPLICATIONS FOR RESEARCHERS AND PRACTITIONERS

The findings from this thesis underline several implications for both researchers and practitioners in the field of requirements engineering. These implications highlight the potential of adopting advanced NLP-based methods for detecting NFRs and point to areas of exploration for future research and practical application.

### 7.3.1 *For Researchers*

The thesis findings present a fertile ground for advancing research in NLP applications for NFR detection.

The success of deep learning methods like convolutional neural networks (CNNs) emphasizes the need to delve deeper into sophisticated linguistic modeling techniques. Hybrid neural architectures that combine CNNs with sequential models, such as Long Short-Term Memory (LSTM) networks or Transformers, represent an exciting area of exploration. While CNNs excel at identifying local patterns within text, LSTMs and Transformers can capture long-term dependencies, enabling a more comprehensive analysis of requirement language. For instance, researchers could investigate the benefits of combining CNNs with Bidirectional LSTMs (BiLSTMs) or integrating them with pre-trained Transformer architectures such as BERT. Such architectures allow for a nuanced understanding of the contextual and structural characteristics of requirements.

The application of contextual word embeddings, such as GloVe, Word2Vec, and FastText, has demonstrated significant promise in improving classification tasks. These embeddings create meaningful vector representations of words based on their context within a corpus, providing valuable insights into semantic relationships. Building on this, researchers may explore embedding techniques that incorporate domain-specific nuances, such as domain-adapted versions of BERT (e.g., SciBERT for scientific texts or FinBERT for financial data). Additionally, multi-task learning can offer an innovative way to share linguistic features across tasks like detecting privacy, fairness, and

security requirements, which could lead to better generalization in data-scarce environments.

Graph-based approaches, like Graph Neural Networks (GNNs), are another promising direction. These models can encode syntactic dependencies and semantic relationships within a graph structure, allowing for richer representation of requirements and their interdependencies. For example, researchers might use Graph Convolutional Networks (GCNs) to model subject-verb-object relationships or even inter-requirement dependencies, thus offering insights into how requirements influence one another.

The findings also highlight the importance of explainability in AI models for requirements engineering. Techniques like SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-Agnostic Explanations) could provide stakeholders with clear justifications for the predictions made by these models. This is particularly critical in domains like healthcare or finance, where decisions based on NFRs must meet strict regulatory standards.

While such research directions focus on the works proposed in this dissertation, we can also think about a wider pipelines exploiting the approaches presented here. For example, in [224], we propose a LLM-based solution to automatically generate requirements starting from conversations between stakeholders and customers. Once such requirements are consolidated, leveraging our approaches is possible to understand which one may have NFR-related concerns and automatically derive quality statements of the system to be developed.

Other ideas could be to take advantage of the approaches during the maintainability of software. In [34], we examined the idea of the automatic generation of the rationale for code changes. The results are negative, since we were able to generate a meaningful motivations only for $\sim$3% of commits. However, futuristically speaking, let's think about a solution that's able to provide a rationale for code changes. Starting from such motivations, we can leverage approaches that can detect NFR to understand how those code changes may have affected the qualities of the software.

### 7.3.2 *For Practitioners*

For practitioners, the findings of this thesis present immediate opportunities to improve the efficiency and accuracy of requirements engineering processes. The implications are equally transformative. The demonstrated effectiveness of Transfer Learning, as seen in the $PD_{TL}$ model for privacy detection, suggests that organizations can adopt pre-trained models to significantly reduce the effort and cost associated with developing domain-specific solutions. These models, fine-tuned on task-specific data, enable practitioners to achieve high performance even with limited datasets. For example, a company developing software for healthcare could use a pre-trained NLP model fine-tuned on privacy-related requirements, accelerating the identification of critical NFRs. Modern computational infrastructures, such as GPUs and TPUs, play a pivotal role in enabling the application of computationally intensive NLP methods. By leveraging cloud-based solutions, organizations can scale their computational capabilities without heavy upfront investment. This accessibility allows smaller teams to adopt state-of-the-art NLP models, democratizing advanced requirements analysis techniques across industries. An example usage of such findings is indeed the web application developed [30], in which the solution we implemented in [31] is exploited and represent the core of the software able to alert requirement engineers regarding privacy concerns into the requirements given as input.

Domain-agnostic NLP technologies offer practitioners a solution for detecting NFRs across various applications. Tools like BERT, trained on general-purpose corpora, can be adapted to identify security, privacy, and fairness requirements. However, the findings underscore the importance of fine-tuning these models for domain-specific contexts to achieve optimal performance. Organizations can build shared repositories of fine-tuned models for different domains, promoting collaboration and standardization across teams.

Interdisciplinary collaboration is another critical area where these findings can drive change. Domain-agnostic tools provide a common language for diverse stakeholders, bridging the gap between technical teams and business analysts. For instance, NLP tools can analyze

user stories to identify NFRs that align with organizational goals, facilitating more effective communication and decision-making.

The results related to fairness-aware NLP methods highlight the ethical responsibilities of practitioners. By integrating fairness checks into their development pipelines, organizations can ensure that software aligns with ethical standards and avoids reinforcing biases. For example, fairness-aware models can flag requirements that inadvertently marginalize certain user groups, enabling proactive mitigation.

Finally, the findings provide actionable strategies for adopting scalable, domain-adaptable, and ethically responsible NLP solutions. Together, these implications point to a future where NLP-driven automation becomes part of requirements engineering, bridging the gap between academic innovation and real-world application.

# 8

# CONCLUSION

The research presented in this dissertation has focused on advancing the automation of NFRs detection by leveraging state-of-the-art NLP technologies and ML and DL techniques. Addressing critical challenges in privacy, security, fairness, and performance, this thesis has aimed to mitigate knowledge gaps in requirements engineering, particularly concerning the implicit and abstract nature of NFRs.

The thesis introduced and validated a series of innovative methodologies. The work on privacy requirements demonstrated how Transfer Learning models can enhance the detection of privacy-related information in agile User Stories by combining syntactic and semantic features with privacy lexicons. This approach not only improved accuracy but also showcased the potential of pre-trained neural networks for domain adaptation. Similarly, the exploration of fairness as a NFR revealed the importance of contextual awareness in recommender systems, where domain-specific biases must be considered to ensure ethical and fair software development.

The work on security requirements highlighted the significant advantages of employing pre-trained Transformer models like BERT, which demonstrated strong domain independence and the ability to generalize across different software contexts. This marks a major step forward in automating the identification of security requirements while ensuring scalability and adaptability. Finally, the investigation into Quantum Natural Language Processing (QNLP) for NFR classification provided valuable insights into emerging technologies, showing promising but limited potential in handling complex linguistic structures, particularly when datasets are small.

This thesis has made substantial contributions to both academia and industry. From an academic perspective, this research has demonstrated the feasibility of leveraging NLP-based approaches for automated NFR detection while exploring the applicability of technologies such as QNLP. Additionally, by systematically evaluating

multiple approaches across diverse NFRs, this work provides an empirical foundation that future studies can build upon, particularly in the intersection of AI and software engineering.

For industry practitioners, the findings serve as a blueprint for integrating NLP-driven NFR detection mechanisms into software development workflows. The demonstrated improvements in accuracy and efficiency across privacy, security, and fairness indicate that organizations can adopt these techniques to enhance the early identification and management of NFRs, reducing the costly design flaws and compliance violations. Moreover, the research highlights how domain-agnostic pre-trained models can be tailored for specific organizational needs, making AI-driven requirement analysis more accessible and adaptable.

During the research, several promising avenues were identified but deemed out of scope due to time and feasibility constraints. One example is the use of Named Entity Recognition (NER) for NFR analysis and elicitation. While traditional classification techniques proved effective in detecting privacy, security, and fairness requirements, NER-based approaches might have offered complementary insights, as been demonstrated when analyzing NFRs such as privacy. Future work could explore the comparative advantages of NER in other contexts and/or NFRs and how it might be integrated into existing NLP pipelines for improving requirement engineering. Additionally, a deeper exploration into multi-modal NFR detection—incorporating visual, structured, and unstructured data—could further enhance automated requirement analysis. Given the increasing complexity of software systems, combining textual analysis with structured metadata, diagrams, or user feedback could yield a more holistic approach to NFR detection.

While significant progress has been made, the findings of this thesis open several avenues for future research. These include further optimization of QNLP models for real-world application, exploring the integration of domain-specific knowledge into pre-trained transformers, and investigating the ethical implications of automated NFR detection systems in sensitive domains. Additionally, expanding NFR datasets and enhancing their annotation quality remain critical steps for improving the generalizability of these techniques.

From another point view, practical studies involving real-world software engineers could be conducted to assess the applicability and usefulness of the proposed NFR classification approaches beyond traditional performance metrics. This could take the form of controlled experiments or case studies, evaluating how practitioners perceive the effectiveness, usability, and integration potential of these methods in real software development workflows.

In conclusion, this dissertation advances the field of requirements engineering by proposing automated, scalable, and adaptable approaches to NFR detection. By bridging gaps in existing methodologies and addressing emerging challenges, it paves the way for a new generation of intelligent RE tools capable of supporting developers in building high-quality, secure, and fair software systems. The lessons learned and tools developed in this research provide a foundation for continued innovation in automating the detection of NFRs, ensuring their integration into the earliest stages of software development.

## ONLINE MATERIAL

**Detecting privacy requirements from User Stories with NLP transfer learning models**

Link to the paper:

https://www.sciencedirect.com/science/article/pii/S0950584922000246

Link to the replication package:

https://tinyurl.com/US-privacy

**Beyond Domain Dependency in Security Requirements Identification**

Link to the replication package:

https://zenodo.org/records/10438323

**Automatic Identification of Privacy and Security Requirements: A Systematic Literature Review**

Link to the replication package:

https://drive.google.com/drive/folders/1OOm55P-iR-nSSpm0PELCMhE8q9--YzcM?usp=sharing

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

Link to the replication package:

https://drive.google.com/drive/u/2/folders/1tULvfFaa2tAJREf2HWpJzpNYvtWx2ntg

**RECOVER: Toward the Automatic Requirements Generation from Stakeholders' Conversations**

Link to the paper:

https://arxiv.org/abs/2411.19552

**PReDUS: A Privacy Requirements Detector From User Stories**

Link to the paper:

https://ceur-ws.org/Vol-3122/NLP4RE-paper-4.pdf

Link to the replication package:

https://github.com/fcasillo/PReDUS-A-Privacy-Requirements-Detector-from-User-Stories

**ReFAIR: Toward a Context-Aware Recommender for Fairness Requirements Engineering**

Link to the paper:

https://dl.acm.org/doi/10.1145/3597503.3639185

Link to the replication package:

https://zenodo.org/records/10470916

**Towards Generating the Rationale for Code Changes**

Link to the paper:

https://antoniomastropaolo.com/assets/pdf/ICPC_RENE_2025.pdf

Link to the replication package:

https://zenodo.org/records/8187207

# BIBLIOGRAPHY


[1] Mina Abbaszade, Mariam Zomorodi, Vahid Salari, and Philip Kurian. "Toward Quantum Machine Translation of Syntactically Distinct Languages." In: *ArXiv* abs/2307.16576 (2023). URL: https://api.semanticscholar.org/CorpusID:260334460.

[2] Muhammad Aurangzeb Ahmad, Carly Eckert, and Ankur Teredesai. "Interpretable machine learning in healthcare." In: *Proceedings of the 2018 ACM international conference on bioinformatics, computational biology, and health informatics*. 2018, pp. 559–560.

[3] Osamah AlDhafer, Irfan Ahmad, and Sajjad Mahmood. "An end-to-end deep learning system for requirements classification using recurrent neural networks." In: *Inf. Softw. Technol.* 147 (2022), p. 106877.

[4] Sousuke Amasaki and Pattara Leelaprute. "The Effects of Vectorization Methods on Non-Functional Requirements Classification." In: *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (2018), pp. 175–182.

[5] David Ameller, Claudia P. Ayala, Jordi Cabot, and Xavier Franch. "How do software architects consider non-functional requirements: An exploratory study." In: *2012 20th IEEE International Requirements Engineering Conference (RE)* (2012), pp. 41–50. URL: https://api.semanticscholar.org/CorpusID:6992459.

[6] Anonymous. *Beyond Domain Dependency in Security Requirements Identification*. 2023. DOI: 10.5281/zenodo.10438323. URL: https://doi.org/10.5281/zenodo.10438323.

[7] Anonymous. *A First Eye on Non-Functional Requirements Detection with Quantum NLP*. Sept. 2024.

[8] P. Anthonysamy, A. Rashid, and R. Chitchyan. "Privacy Requirements: Present Future." In: *Proceedings of IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Society Track (ICSE-SEIS)*. 2017, pp. 13–22. DOI: 10.1109/ICSE-SEIS.2017.3.

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

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

[11] Ken Barker, Mina Askari, Mishtu Banerjee, Kambiz Ghazinour, Brenan Mackas, Maryam Majedi, Sampson Pun, and Adepele Williams. "A Data Privacy Taxonomy." In: *Proceedings of the 26th British National Conference on Databases: Dataspace: The Final Frontier*. BNCOD 26. Birmingham, UK: Springer-Verlag, 2009, 42–54. ISBN: 9783642028427. DOI: 10.1007/978-3-642-02843-4_7.

[12] Solon Barocas, Moritz Hardt, and Arvind Narayanan. "Fairness in machine learning." In: *Nips tutorial* 1 (2017), p. 2.

[13] Richard Berntsson-Svensson, Tony Gorschek, and Björn Regnell. "Quality Requirements in Practice: An Interview Study in Requirements Engineering for Embedded Systems." In: *Requirements Engineering: Foundation for Software Quality*. 2009.

[14] Dane Bertram. "Likert scales." In: *Retrieved November* 2.10 (2007), pp. 1–10.

[15] Seblewongel Esseynew Biable, Nuno Manuel Garcia, Dida Midekso, and Nuno Pombo. "Ethical Issues in Software Requirements Engineering." In: *Software* 1.1 (2022), pp. 31–52. ISSN: 2674-113X.

[16] Manal Binkhonain and Liping Zhao. "A review of machine learning algorithms for identification and classification of non-functional requirements." In: *Expert Syst. Appl. X* 1 (2019).

[17] Sumon Biswas and Hridesh Rajan. "Fair Preprocessing: Towards Understanding Compositional Fairness of Data Transformers in Machine Learning Pipeline." In: *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2021. Athens, Greece: Association for Computing Machinery, 2021, 981–993. ISBN: 9781450385626.

[18] B. Boehm and V.R. Basili. "Top 10 list software development." In: *Computer* 34.1 (2001), pp. 135–137.

[19] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. "Enriching Word Vectors with Subword Information." In: *arXiv preprint arXiv:1607.04606* (2016).

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

[21] Andreas Borg, Angela Yong, Pär Carlshamre, and Kristian Sandahl. "The Bad Conscience of Requirements Engineering : An Investigation in Real-World Treatment of Non-Functional Requirements." In: *Computer Science* (2003). URL: https://api.semanticscholar.org/CorpusID:12513811.

[22] Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. "Large Language Models in Machine Translation." In: *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Prague, Czech Republic: Association for Computational Linguistics, 2007, pp. 858–867. URL: https://aclanthology.org/D07-1090.

[23] Leo Breiman. "Bagging predictors." In: *Machine Learning* (1996).

[24] Frederick P. Brooks. "No Silver Bullet: Essence and Accidents of Software Engineering." In: 1987.

[25] Bernd Bruegge and Allen H Dutoit. *Object-Oriented Software Engineering Using UML, Patterns, and Java™ Third Edition*. by Pearson Education, Inc., 2010.

[26] Yuriy Brun and Alexandra Meliou. "Software fairness." In: *Proceedings of the 2018 26th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering*. 2018, pp. 754–759.

[27] Marc-Etienne Brunet, Colleen Alkalay-Houlihan, Ashton Anderson, and Richard Zemel. "Understanding the Origins of Bias in Word Embeddings." In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 803–811.

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

[29] Francesco Casillo, Vincenzo Deufemia, and Carmine Gravino. "Detecting Privacy Requirements from User Stories with NLP Transfer Learning Models." In: *Inf. Softw. Technol.* 146 (2022).

[30] Francesco Casillo, Vincenzo Deufemia, and Carmine Gravino. "PReDUS: A Privacy Requirements Detector From User Stories." In: *REFSQ Workshops*. 2022.

[31] Francesco Casillo, Vincenzo Deufemia, and Carmine Gravino. "**Detecting privacy requirements from User Stories with NLP transfer learning models**." In: *Information and Software Technology* 146 (2022), p. 106853. ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2022.106853.

[32] Francesco Casillo, Vincenzo Deufemia, and Carmine Gravino. "**Automatic Identification of Privacy and Security Requirements: A Systematic Literature Review**." In: *Under minor revision in Requirement Engineering Journal* (2025).

[33] Francesco Casillo, Vincenzo Deufemia, and Carmine Gravino. "**Beyond Domain Dependency in Security Requirements Identification**." In: *Information and Software Technology*, 182 (2025), p. 107702. ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2025.107702.

[34] Francesco Casillo, Antonio Mastropaolo, Gabriele Bavota, Vincenzo Deufemia, and Carmine Gravino. "Towards Generating the Rationale for Code Changes." In: *33rd IEEE/ACM International Conference on Program Comprehension (ICPC 2025)*. ICSE '25. Ottawa, Canada, 2025.

[35] L. Elisa Celis, Lingxiao Huang, Vijay Keswani, and Nisheeth K. Vishnoi. "Classification with Fairness Constraints: A Meta-Algorithm with Provable Guarantees." In: *Proceedings of the Conference on Fairness, Accountability, and Transparency*. FAT* '19. Atlanta, GA, USA: Association for Computing Machinery, 2019, 319–328. ISBN: 9781450361255.

[36] Joymallya Chakraborty, Suvodeep Majumder, and Tim Menzies. "Bias in machine learning software: why? how? what to do?" In: *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2021, pp. 429–440.

[37] Kai-Wei Chang, Vinodkumar Prabhakaran, and Vicente Ordonez. "Bias and fairness in natural language processing." In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): Tutorial Abstracts*. 2019.

[38] Francis Chantree. "Identifying Nocuous Ambiguities in Natural Language Requirements." In: *14th IEEE International Requirements Engineering Conference (RE'06)* (2006), pp. 59–68.

[39] Ranit Chatterjee, Abdul Ahmed, Preethu Rose Anish, Brijendra Kumar Suman, Prashant Lawhatre, and Smita Ghaisas. "A Pipeline for Automating Labeling to Prediction in Classification of NFRs." In: *2021 IEEE 29th International Requirements Engineering Conference (RE)* (2021), pp. 323–323.

[40] Cheligeer Cheligeer, Jingwei Huang, Guosong Wu, Nadia Bhuiyan, Yuan Xu, and Yong Zeng. "Machine learning in requirements elicitation: a literature review." In: *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 36 (2022), e32.

[41] Tianqi Chen and Carlos Guestrin. "XGBoost." In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016. DOI: 10.1145/2939672.2939785. URL: https://doi.org/10.1145\%2F2939672.2939785.

[42] Zhenpeng Chen, Jie M. Zhang, Federica Sarro, and Mark Harman. "MAAT: A Novel Ensemble Approach to Addressing Fairness and Performance Bugs for Machine Learning Software." In: *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2022. Singapore, Singapore: Association for Computing Machinery, 2022, 1122–1134. ISBN: 9781450394130.

[43] Alexandra Chouldechova and Aaron Roth. "A snapshot of the frontiers of fairness in machine learning." In: *Communications of the ACM* 63.5 (2020), pp. 82–89.

[44] Lawrence Chung, Brian A Nixon, Eric Yu, and John Mylopoulos. *Non-functional requirements in software engineering*. Vol. 5. Springer Science & Business Media, 2012.

[45] Stephen Clark. "Something Old, Something New: Grammar-based CCG Parsing with Transformer Models." In: *ArXiv* abs/2109.10044 (2021). URL: https://api.semanticscholar.org/CorpusID:237581168.

[46] Jane Cleland-Huang, Raffaella Settimi, Xuchang Zou, and Peter Solc. "The Detection and Classification of Non-Functional Requirements with Application to Early Aspects." In: *14th IEEE International Requirements Engineering Conference (RE'06)* (2006), pp. 39–48.

[47] Jane Cleland-Huang, Raffaella Settimi, Xuchang Zou, and Peter Solc. "The Detection and Classification of Non-Functional Requirements with Application to Early Aspects." In: *14th IEEE International Requirements Engineering Conference (RE'06)* (2006), pp. 39–48.

[48] Jane Cleland-Huang, Raffaella Settimi, Xuchang Zou, and Peter Solc. "Automated Classification of Non-Functional Require-

ments." In: *Requir. Eng.* 12.2 (2007), 103–120. ISSN: 0947-3602. DOI: 10.1007/s00766-007-0045-1.

[49] Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. "Mathematical Foundations for a Compositional Distributional Model of Meaning." In: *ArXiv* abs/1003.4394 (2010).

[50] Mike Cohn. *User Stories Applied: For Agile Software Development*. Addison Wesley, 2004. ISBN: 0-321-20568-5.

[51] Mike Cohn. *User stories applied: For agile software development*. Addison-Wesley Professional, 2004.

[52] William Jay Conover. *Practical nonparametric statistics*. Wiley series in probability and statistics. Wiley, 1999.

[53] F. Dalpiaz. *Requirements data sets (user stories)*. https://data.mendeley.com/dat 2018. DOI: 10.17632/7zbk8zsd8y.1.

[54] Fabiano Dalpiaz, Ivor Van Der Schalk, Sjaak Brinkkemper, Fatma Basak Aydemir, and Garm Lucassen. "Detecting terminological ambiguity in user stories: Tool and experimentation." In: *Inf. Softw. Technol.* 110 (2019), pp. 3–16.

[55] S. De Capitani Di Vimercati, S. Foresti, G. Livraga, and P. Samarati. "Data Privacy: Definitions and Techniques." In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 20.06 (2012), pp. 793–817. DOI: 10.1142/S0218488512400247.

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

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

[58] J.L. Devore, N.R. Farnum, and J.A. Doi. *Applied Statistics for Engineers and Scientists*. Cengage Learning, 2013. ISBN: 9781133111368. URL: https://books.google.it/books?id=psg_CQAAQBAJ.

[59] Edna Dias Canedo and Bruno Cordeiro Mendes. "Software Requirements Classification Using Machine Learning Algorithms." In: *Entropy* 22.9 (2020). ISSN: 1099-4300.

[60] Mengnan Du, Fan Yang, Na Zou, and Xia Hu. "Fairness in deep learning: A computational perspective." In: *IEEE Intelligent Systems* 36.4 (2020), pp. 25–34.

[61] Nicolau Duran-Silva, Enric Fuster, Francesco Alessandro Massucci, César Parra-Rojas, Arnau Quinquillà, Fernando Roda, Bernardo Rondelli, Nicandro Bovenzi, and Chiara Toietta. *A controlled vocabulary for research and innovation in the field of Artificial Intelligence (AI)*. en. 2021. DOI: 10.5281/ZENODO.5591987. URL: https://zenodo.org/record/5591987.

[62] Fahime Ebrahimi and Anas Mahmoud. "Unsupervised Summarization of Privacy Concerns in Mobile Application Reviews." In: *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering* (2022).

[63] Meryem Elallaoui, Khalid Nafil, and Raja Touahni. "Automatic Transformation of User Stories into UML Use Case Diagrams using NLP Techniques." In: *Procedia Computer Science* 130 (2018). The 9th International Conference on Ambient Systems, Networks and Technologies (ANT 2018) / The 8th International Conference on Sustainable Energy Information Technology (SEIT-2018) / Affiliated Workshops, pp. 42–49. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2018.04.010.

[64] Alessandro Fabris, Stefano Messina, Gianmaria Silvello, and Gian Antonio Susto. "Algorithmic fairness datasets: the story so far." In: *Data Mining and Knowledge Discovery* 36.6 (2022), pp. 2074–2152. DOI: 10.1007/s10618-022-00854-z. URL: https://doi.org/10.1007/s10618-022-00854-z.

[65] Zhangyin Feng et al. "CodeBERT: A Pre-Trained Model for Programming and Natural Languages." In: *CoRR* abs/2002.08155 (2020). arXiv: 2002.08155. URL: https://arxiv.org/abs/2002.08155.

[66] D. Méndez Fernández et al. "Naming the pain in requirements engineering - Contemporary problems, causes, and effects in practice." Version 22(5). In: *Empirical software engineering* (2017), pp. 2298–2338. DOI: 10.1007/s10664-016-9451-7.

[67] Carmine Ferrara, Francesco Casillo, Carmine Gravino, Andrea De Lucia, and Fabio Palomba. *ReFAIR: Toward a Context-Aware Recommender for Fairness Requirements Engineering*. New York, NY, USA, 2024. DOI: 10.1145/3597503.3639185. URL: https://doi.org/10.1145/3597503.3639185.

[68] Alessio Ferrari, Giorgio Oronzo Spagnolo, and Stefania Gnesi. *PURE: a Dataset of Public Requirements Documents*. Version 2.0. Sept. 2022. DOI: 10.5281/zenodo.7118517. URL: https://doi.org/10.5281/zenodo.7118517.

[69] Anthony Finkelstein, Mark Harman, S Afshin Mansouri, Jian Ren, and Yuanyuan Zhang. ""Fairness analysis" in requirements assignments." In: *2008 16th IEEE International Requirements Engineering Conference*. IEEE. 2008, pp. 115–124.

[70] Sainyam Galhotra, Yuriy Brun, and Alexandra Meliou. "Fairness testing: testing software for discrimination." In: *Proceedings of the 2017 11th Joint meeting on foundations of software engineering*. 2017, pp. 498–510.

[71] Stefan Gärtner, Thomas Ruhroth, Jens Bürger, Kurt Schneider, and Jan Jürjens. "Maintaining requirements for long-living software systems by incorporating security knowledge." In: *2014 IEEE 22nd International Requirements Engineering Conference (RE)* (2014), pp. 103–112.

[72] A. Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019. ISBN: 9781492032595. URL: https://books.google.ch/books?id=HnetDwAAQBAJ.

[73] Alastair J. Gill, Asimina Vasalou, Chrysanthi Papoutsi, and Adam N. Joinson. "Privacy Dictionary: A Linguistic Taxonomy of Privacy for Content Analysis." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, 2011, 3227–3236. ISBN: 9781450302289. DOI: 10.1145/1978942.1979421.

[74] R. Guarasci, Giuseppe De Pietro, and Massimo Esposito. "Quantum Natural Language Processing: Challenges and Opportuni-

ties." In: *Applied Sciences* (2022). URL: https://api.semanticscholar.org/CorpusID:249333405.

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

[76] Vojtěch Havlíček, Antonio D. Córcoles, Kristan Temme, Aram Wettroth Harrow, Abhinav Kandala, Jerry M. Chow, and Jay M. Gambetta. "Supervised learning with quantum-enhanced feature spaces." In: *Nature* 567 (2018), pp. 209 –212. URL: https://api.semanticscholar.org/CorpusID:51680972.

[77] Petra Heck and Andy Zaidman. *A Quality Framework for Agile Requirements: A Practitioner's Perspective*. 2014. arXiv: 1406.4692 [cs.SE].

[78] Valery Herrington1, Oluwafemi Adedeji2, and Valery Herrington. "At The Intersection of Medical Robotic Surgery and Drug Discovery with Quantum Computing." In: *Journal of Electrical Electronics Engineering* (2023). URL: https://api.semanticscholar.org/CorpusID:261157948.

[79] Tobias Hey, Jan Keim, Anne Koziolek, and Walter F. Tichy. "NoRBERT: Transfer Learning for Requirements Classification." In: *2020 IEEE 28th International Requirements Engineering Conference (RE)* (2020), pp. 169–179.

[80] Frederic R. Hopp, Jacob T. Fisher, Devin Cornell, Richard Huskey, and René Weber. "The extended Moral Foundations Dictionary (eMFD): Development and applications of a crowd-sourced approach to extracting moral intuitions from text." In: *Behavior Research Methods* 53.1 (2020), pp. 232–246.

[81] Max Hort, Jie M. Zhang, Federica Sarro, and Mark Harman. "Fairea: A Model Behaviour Mutation Approach to Benchmarking Bias Mitigation Methods." In: *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*.

ESEC/FSE 2021. Athens, Greece: Association for Computing Machinery, 2021, 994–1006. ISBN: 9781450385626.

[82] Xiaolei Huang, Linzi Xing, Franck Dernoncourt, and Michael J. Paul. "Multilingual Twitter Corpus and Baselines for Evaluating Demographic Bias in Hate Speech Recognition." English. In: *Proceedings of the Twelfth Language Resources and Evaluation Conference*. European Language Resources Association, 2020, pp. 1440–1448. ISBN: 979-10-95546-34-4.

[83] Sayem Mohammad Imtiaz, Md Rayhan Amin, Anh Quoc Do, Stefano Iannucci, and Tanmay Bhowmik. "Predicting Vulnerability for Requirements." In: *2021 IEEE 22nd International Conference on Information Reuse and Integration for Data Science (IRI)* (2021), pp. 160–167.

[84] Sayem Mohammad Imtiaz and Tanmay Bhowmik. "Towards data-driven vulnerability prediction for requirements." In: *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (2018).

[85] Irum Inayat, Siti Salwah Salim, Sabrina Marczak, Maya Daneva, and Shahaboddin Shamshirband. "A systematic literature review on agile requirements engineering practices and challenges." In: *Computers in human behavior* 51 (2015), pp. 915–929.

[86] Neil Ireson, Fabio Ciravegna, Mary Elaine Califf, Dayne Freitag, Nicholas Kushmerick, and Alberto Lavelli. "Evaluating Machine Learning for Information Extraction." In: *ICML '05: Proceedings of the 22nd International Conference on Machine Learning*. ICML '05. New York, NY, USA: ACM, 2005, 345–352. ISBN: 1595931805. DOI: 10.1145/1102351.1102395.

[87] Neil Ireson, Fabio Ciravegna, Mary Elaine Califf, Dayne Freitag, Nicholas Kushmerick, and Alberto Lavelli. "Evaluating Machine Learning for Information Extraction." In: *ICML '05: Proceedings of the 22nd International Conference on Machine Learning*. ICML '05. New York, NY, USA: ACM, 2005, 345–352. ISBN: 1595931805. DOI: 10.1145/1102351.1102395.

[88] Vladimir Ivanov, Andrey Sadovykh, Alexandr Naumchev, Alessandra Bagnato, and Kirill Yakovlev. *Extracting Software Requirements from Unstructured Documents*. 2022. arXiv: 2202.02135 [cs.SE].

[89] Nathalie Japkowicz and Mohak Shah. *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge University Press, 2011. DOI: 10.1017/CBO9780511921803.

[90] S. Jiménez and R. Juárez-Ramírez. "A Quality Framework for Evaluating Grammatical Structure of User Stories to Improve External Quality." In: *Proceedings of 7th International Conference in Software Engineering Research and Innovation (CONISOFT)*. 2019, pp. 147–153. DOI: 10.1109/CONISOFT.2019.00029.

[91] Rajni Jindal, Ruchika Malhotra, and Abha Jain. "Automated classification of security requirements." In: *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (2016), pp. 2027–2033.

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

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

[94] Ammar Ismael Kadhim. "Survey on supervised machine learning techniques for automatic text classification." In: *Artificial Intelligence Review* 52.1 (2019), pp. 273–292.

[95] Wahiba Ben Abdessalem Karaa, Zeineb Ben Azzouz, Aarti Singh, Nilanjan Dey, Amira S. Ashour, and Henda Ben Ghézala. "Automatic builder of class diagram (ABCD): an application of UML generation from functional requirements." In: *Softw. Pract. Exp.* 46.11 (2016), pp. 1443–1458. DOI: 10.1002/spe.2384.

[96] Dimitri Kartsaklis. "Coordination in Categorical Compositional Distributional Semantics." In: *SLPCS@QPL*. 2016. URL: https://api.semanticscholar.org/CorpusID:10842035.

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

[98] Kamaljit Kaur and Parminder Kaur. "SABDM: A self-attention based bidirectional-RNN deep model for requirements classification." In: *Journal of Software: Evolution and Process* (2022).

[99] Kamaljit Kaur and Parminder Kaur. "BERT-CNN: Improving BERT for Requirements Classification using CNN." In: *Procedia Computer Science* 218 (2023). International Conference on Machine Learning and Data Engineering, pp. 2604–2611. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2023.01.234. URL: https://www.sciencedirect.com/science/article/pii/S187705092300234X.

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

[101] Fatemeh Khayashi, Behnaz Jamasb, Reza Akbari, and Pirooz Shamsinejadbabaki. "Deep Learning Methods for Software Requirement Classification: A Performance Study on the PURE dataset." In: *ArXiv* abs/2211.05286 (2022).

[102] Iqra Khurshid, Salma Imtiaz, Wadii Boulila, Zahid Khan, Almas Abbasi, Abdul Rehman Javed, and Zunera Jalil. "Classification of Non-Functional Requirements From IoT Oriented Healthcare Requirement Document." In: *Frontiers in Public Health* 10 (2022).

[103] Eric Knauss, Siv Hilde Houmb, Kurt Schneider, Shareeful Islam, and Jan Jürjens. "Supporting Requirements Engineers in Recognising Security Issues." In: *Requirements Engineering: Foundation for Software Quality*. 2011.

[104] Eric Knauss, Siv Houmb, Kurt Schneider, Shareeful Islam, and Jan Jürjens. "Supporting Requirements Engineers in Recognising Security Issues." In: *Requirements Engineering: Foundation for Software Quality*. Ed. by Daniel Berry and Xavier Franch. Springer Berlin Heidelberg, 2011, pp. 4–18. ISBN: 978-3-642-19858-8.

[105] Eric Knauss, Siv Houmb, Kurt Schneider, Shareeful Islam, and Jan Jürjens. "Supporting Requirements Engineers in Recognising Security Issues." In: *Requirements Engineering: Foundation for Software Quality*. Ed. by Daniel Berry and Xavier Franch. Springer Berlin Heidelberg, 2011, pp. 4–18. ISBN: 978-3-642-19858-8.

[106] Armin Kobilica, Mohammed Ayub, and Jameleddine Hassine. "Automated Identification of Security Requirements: A Machine Learning Approach." In: *Proceedings of the Evaluation and Assessment in Software Engineering* (2020).

[107] Ekrem Kocaguneli, Tim Menzies, and Emilia Mendes. "Transfer learning in effort estimation." In: *Empir. Softw. Eng.* 20.3 (2015), pp. 813–843. DOI: 10.1007/s10664-014-9300-5.

[108] Rahul Krishna and Tim Menzies. "Bellwethers: A Baseline Method for Transfer Learning." In: *IEEE Trans. Software Eng.* 45.11 (2019), pp. 1081–1105. DOI: 10.1109/TSE.2018.2821670.

[109] Zijad 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.

[110] Z. Kurtanović and W. Maalej. "Automatically Classifying Functional and Non-functional Requirements Using Supervised Machine Learning." In: *Proceedings of IEEE 25th International Requirements Engineering Conference (RE)*. 2017, pp. 490–495. DOI: 10.1109/RE.2017.82.

[111] Zijad Kurtanović and Walid 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.

[112] J. Lambek. "Type Grammar Revisited." In: *Logical Aspects of Computational Linguistics*. Ed. by Alain Lecomte, François Lamarche, and Guy Perrier. Springer Berlin Heidelberg, 1999, pp. 1–27. ISBN: 978-3-540-48975-7.

[113] Omer Levy, Yoav Goldberg, and Ido Dagan. "Improving distributional similarity with lessons learned from word embeddings." In: *Transactions of the association for computational linguistics* 3 (2015), pp. 211–225.

[114] Bing chuan Li and Xiuwen Nong. "Automatically classifying non-functional requirements using deep neural network." In: *Pattern Recognit.* 132 (2022), p. 108948.

[115] Chuanyi Li, LiGuo Huang, Jidong Ge, Bin Luo, and Vincent Ng. "Automatically classifying user requests in crowdsourcing requirements engineering." In: *J. Syst. Softw.* 138 (2018), pp. 108–123.

[116] Gang Li, Chengpeng Zheng, Min Li, and Haosen Wang. "Automatic Requirements Classification Based on Graph Attention Network." In: *IEEE Access* 10 (2022), pp. 30080–30090.

[117] Tong Li and Zhishuai Chen. "An ontology-based learning approach for automatically classifying security requirements." In: *Journal of Systems and Software* 165 (2020), p. 110566. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2020.110566.

[118] Tong Li and Zhishuai Chen. "An ontology-based learning approach for automatically classifying security requirements." In: *J. Syst. Softw.* 165 (2020), p. 110566.

[119] Ze Shi Li, Manish Sihag, Nowshin Nawar Arony, Joao Bezerra Junior, Thanh Binh Phan, Neil A. Ernst, and Daniela E. Herlea Damian. "Narratives: the Unforeseen Influencer of Privacy Concerns." In: *2022 IEEE 30th International Requirements Engineering Conference (RE)* (2022), pp. 127–139.

[120] Márcia Lima, Victor Valle, Estevão Costa, Fylype Lira, and Bruno Gadelha. "Software Engineering Repositories: Expanding the PROMISE Database." In: *Proceedings of the XXXIII*

*Brazilian Symposium on Software Engineering*. SBES '19. New York, NY, USA: ACM, 2019, 427–436. ISBN: 9781450376518. DOI: 10.1145/3350768.3350776. URL: https://doi.org/10.1145/3350768.3350776.

[121] Márcia Lima, Victor Valle, Estevão Costa, Fylype Lira, and Bruno F. Gadelha. "Software Engineering Repositories: Expanding the PROMISE Database." In: *Proceedings of the XXXIII Brazilian Symposium on Software Engineering* (2019). URL: https://api.semanticscholar.org/CorpusID:202728524.

[122] Olga Liskin, Raphael Pham, Stephan Kiesling, and Kurt Schneider. "Why We Need a Granularity Concept for User Stories." In: *Agile Processes in Software Engineering and Extreme Programming*. Berlin, Heidelberg: Springer-Verlag, 2014, 110–125. ISBN: 9783642206771.

[123] Delmer Alejandro López-Hernández, Efrén Mezura-Montes, Jorge Octavio Ocharán-Hernández, and Ángel Juan Sánchez-García. "Non-functional Requirements Classification using Artificial Neural Networks." In: *2021 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)* 5 (2021), pp. 1–6.

[124] Delmer Alejandro López-Hernández, Jorge Octavio Ocharán-Hernández, Efrén Mezura-Montes, and Ángel Juan Sánchez-García. "Automatic Classification of Software Requirements using Artificial Neural Networks: A Systematic Literature Review." In: *2021 9th International Conference in Software Engineering Research and Innovation (CONISOFT)* (2021), pp. 152–160.

[125] Nicholas Lourie, Ronan Le Bras, and Yejin Choi. "Scruples: A corpus of community ethical judgments on 32,000 real-life anecdotes." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 2021, pp. 13470–13479.

[126] Oscar Luaces, Jorge Díez, Jose Barranquero, Juan del Coz, and Antonio Bahamonde. "Binary relevance efficacy for multilabel classification." In: *Progress in Artificial Intelligence* 1 (2012). DOI: 10.1007/s13748-012-0030-x.

[127] G. Lucassen, F. Dalpiaz, J. M. van der Werf, and S. Brinkkemper. "Forging high-quality User Stories: Towards a discipline for Agile Requirements." In: *Proceedings of IEEE 23rd International Requirements Engineering Conference (RE)*. 2015, pp. 126–135. DOI: 10.1109/RE.2015.7320415.

[128] Garm Lucassen, Fabiano Dalpiaz, Jan Martijn E. M. van der Werf, and Sjaak Brinkkemper. "The Use and Effectiveness of User Stories in Practice." In: *Requirements Engineering: Foundation for Software Quality*. 2016.

[129] Garm Lucassen, Fabiano Dalpiaz, Jan Martijn van der Werf, and Sjaak Brinkkemper. "The Use and Effectiveness of User Stories in Practice." In: *Requirements Engineering: Foundation for Software Quality*. Ed. by Maya Daneva and Oscar Pastor. Cham: Springer International Publishing, 2016, pp. 205–222.

[130] Garm Lucassen, Marcel Robeer, Fabiano Dalpiaz, Jan Martijn E. M. van der Werf, and Sjaak Brinkkemper. "Extracting conceptual models from user stories with Visual Narrator." In: *Requir. Eng.* 22.3 (2017), pp. 339–358. DOI: 10.1007/s00766-017-0270-1.

[131] Xianchang Luo, Yinxing Xue, Zhenchang Xing, and Jiamou Sun. "PRCBERT: Prompt Learning for Requirement Classification using BERT-based Pretrained Language Models." In: *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering* (2022).

[132] Thang Luong, Hieu Pham, and Christopher D. Manning. "Effective Approaches to Attention-based Neural Machine Translation." In: *ArXiv* abs/1508.04025 (2015).

[133] Milda Maciejauskaitė and Jolanta Miliauskaitė. "THE EFFICIENCY OF MACHINE LEARNING ALGORITHMS IN CLASSIFYING NON-FUNCTIONAL REQUIREMENTS." In: *New Trends in Computer Sciences* (2024). URL: https://api.semanticscholar.org/CorpusID:270644801.

[134] Anas Mahmoud and Grant Williams. "Detecting, classifying, and tracing non-functional software requirements." In: *Requirements Engineering* 21 (2016), pp. 357–381.

[135] Richard R. Maiti and Frank 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.

[136] Vincenzo De Martino and Fabio Palomba. "Classification, Challenges, and Automated Approaches to Handle Non-Functional Requirements in ML-Enabled Systems: A Systematic Literature Review." In: *ArXiv* abs/2311.17483 (2023).

[137] Aaron K. Massey, Jacob Eisenstein, Annie I. Antón, and Peter P. Swire. "Automated text mining for requirements analysis of policy documents." In: *2013 21st IEEE International Requirements Engineering Conference (RE)* (2013), pp. 4–13.

[138] Stuart McIlroy, Nasir Ali, Hammad Khalid, and A. Hassan. "Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews." In: *Empirical Software Engineering* 21 (2015), pp. 1067–1106.

[139] Wes McKinney. "Data Structures for Statistical Computing in Python." In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman. 2010, pp. 56 –61. DOI: 10.25080/Majora-92bf1922-00a.

[140] Nuhil Mehdy, Casey Kennington, and Hoda Mehrpouyan. "Privacy Disclosures Detection in Natural-Language Text Through Linguistically-Motivated Artificial Neural Networks." In: *Security and Privacy in New Computing Environments*. 2019, pp. 152–177. DOI: 10.1007/978-3-030-21373-2_14.

[141] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. "A Survey on Bias and Fairness in Machine Learning." In: *ACM Comput. Surv.* 54.6 (2021). ISSN: 0360-0300.

[142] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. "A survey on bias and fairness in machine learning." In: *ACM Computing Surveys (CSUR)* 54.6 (2021), pp. 1–35.

[143] Marina Meilă. "Comparing clusterings—an information based distance." In: *Journal of Multivariate Analysis* 98.5 (2007), pp. 873–895. ISSN: 0047-259X. DOI: https://doi.org/10.1016/j.jmva.2006.11.013. URL: https://www.sciencedirect.com/science/article/pii/S0047259X06002016.

[144] Mahmuda Akter Metu, Nazneen Akhter, Sanjeda Nasrin, Tasnim Anzum, Afrina Khatun, and Rashed Mazumder. "Hybrid SVM-Bidirectional Long Short-Term Memory Model for Fine-Grained Software Requirement Classification." In: *Journal of Advances in Information Technology* (2024).

[145] Ahmed Metwally and Chun-Heng Huang. "Scalable Similarity Joins of Tokenized Strings." In: *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. 2019, pp. 1766–1777. DOI: 10.1109/ICDE.2019.00193.

[146] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL].

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

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

[149] Claire Cain Miller. "Can an algorithm hire better than a human." In: *The New York Times* 25 (2015).

[150] Mazen Mohamad, Jan-Philipp Steghöfer, Alexander Åström, and Riccardo Scandariato. "Identifying security-related requirements in regulatory documents based on cross-project classification." In: *Proceedings of the 18th International Conference on Predictive Models and Data Analytics in Software Engineering* (2022).

[151] Mazen Mohamad, Jan-Philipp Steghöfer, Alexander Åström, and Riccardo Scandariato. "Identifying Security-Related Requirements in Regulatory Documents Based on Cross-Project Classification." In: *PROMISE 2022*. New York, NY, USA: ACM, 2022, 82–91. ISBN: 9781450398602. DOI: 10.1145/3558489.3559074. URL: https://doi.org/10.1145/3558489.3559074.

[152] Zafeiria Moumoulidou, Andrew McGregor, and Alexandra Meliou. "Diverse Data Selection under Fairness Constraints." In: *arXiv preprint arXiv:2010.09141* (2020).

[153] Nuthan Munaiah, Andrew Meneely, and Pradeep K. Murukannaiah. "A Domain-Independent Model for Identifying Security Requirements." In: *2017 IEEE 25th International Requirements Engineering Conference (RE)* (2017), pp. 506–511.

[154] Nuthan Munaiah, Andrew Meneely, and Pradeep K Murukannaiah. "A domain-independent model for identifying security requirements." In: *2017 IEEE 25th International Requirements Engineering Conference (RE)*. IEEE. 2017, pp. 506–511.

[155] Fatemeh Nargesian, Abolfazl Asudeh, and H. V. Jagadish. "Responsible Data Integration: Next-Generation Challenges." In: *Proceedings of the 2022 International Conference on Management of Data*. SIGMOD '22. Philadelphia, PA, USA: Association for Computing Machinery, 2022, 2458–2464. ISBN: 9781450392495.

[156] J. Neerbeky, I. Assentz, and P. Dolog. "TABOO: Detecting Unstructured Sensitive Information Using Recursive Neural Networks." In: *Proceedings of IEEE 33rd International Conference on Data Engineering (ICDE)*. 2017, pp. 1399–1400. DOI: 10.1109/ICDE.2017.195.

[157] Duc Cuong Nguyen, Erik Derr, Michael Backes, and Sven Bugiel. "Short Text, Large Effect: Measuring the Impact of User Reviews on Android App Security & Privacy." In: *2019 IEEE Symposium on Security and Privacy (SP)* (2019), pp. 555–569.

[158] Quyen L. Nguyen. "Non-functional requirements analysis modeling for software product lines." In: *2009 ICSE Workshop on Modeling in Software Engineering*. 2009, pp. 56–61.

[159] Quyen L. Nguyen. "Non-functional requirements analysis modeling for software product lines." In: *Proceedings of ICSE Workshop on Modeling in Software Engineering, MiSE 2009, Vancouver, BC, Canada, May 17-18, 2009*. IEEE Computer Society, 2009, pp. 56–61. DOI: 10.1109/MISE.2009.5069898.

[160] Laura Okpara, Colin Werner, Adam Murray, and Daniela Damian. "The role of informal communication in building shared understanding of non-functional requirements in remote continuous software engineering." In: *Requirements Engineering* 28.4 (2023), pp. 595–617.

[161] Michel Oleynik, Amila Kugic, Zdenko Kasáč, and Markus Kreuzthaler. "Evaluating shallow and deep learning strategies for the 2018 n2c2 shared task on clinical text classification." In: *Journal of the American Medical Informatics Association* 26.11 (2019), pp. 1247–1254. ISSN: 1527-974X. DOI: 10.1093/jamia/ocz149.

[162] Anderson Oliveira, João Lucas Correia, Wesley K. G. Assunção, Juliana Alves Pereira, Rafael Maiani de Mello, Daniel Coutinho, Caio Barbosa, Paulo Libório, and Alessandro Garcia. "Understanding Developers' Discussions and Perceptions on Non-functional Requirements: The Case of the Spring Ecosystem." In: *Proc. ACM Softw. Eng.* 1 (2024), pp. 517–538. URL: https://api.semanticscholar.org/CorpusID:270869076.

[163] Parmy Olson. "The algorithm that beats your bank manager." In: *CNN Money March* 15 (2011).

[164] Carla L. Pacheco, Iván A. García, and Miryam Reyes. "Requirements elicitation techniques: a systematic literature review based on the maturity of the techniques." In: *IET Softw.* 12 (2018), pp. 365–378.

[165] F. Paetsch, A. Eberlein, and F. Maurer. "Requirements engineering and agile software development." In: *WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003.* 2003, pp. 308–313.

[166] Frauke Paetsch, Armin Eberlein, and Frank Maurer. "Requirements Engineering and Agile Software Development." In: *Proceedings of 12th IEEE International Workshops on Enabling Technologies (WETICE 2003), Infrastructure for Collaborative Enterprises, 9-11 June 2003, Linz, Austria*. IEEE Computer Society, 2003, pp. 308–313. DOI: 10.1109/ENABL.2003.1231428.

[167] Krupa Patel and Unnati Shah. "Requirements Specification." In: *The Future of Artificial Intelligence and Robotics: Proceedings of 5th International Conference on Deep Learning, Artificial Intelligence and Robotics (ICDLAIR)*. Vol. 1001. Springer Nature. 2023, p. 147.

[168] Fabian Pedregosa et al. "Scikit-learn: Machine Learning in Python." In: *Journal of Machine Learning Research* 12.85 (2011), pp. 2825–2830. URL: http://jmlr.org/papers/v12/pedregosa11a.html.

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

[170] Jeffrey Pennington, Richard Socher, and Christopher Manning. "GloVe: Global Vectors for Word Representation." In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://aclanthology.org/D14-1162.

[171] Dana Pessach and Erez Shmueli. "A Review on Fairness in Machine Learning." In: *ACM Comput. Surv.* 55.3 (2022). ISSN: 0360-0300. DOI: 10.1145/3494672. URL: https://doi.org/10.1145/3494672.

[172] Klaus Pohl. *Requirements Engineering: Fundamentals, Principles, and Techniques*. 1st. Springer Publishing Company, Incorporated, 2010. ISBN: 3642125778.

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

[174] David M. W. Powers. *Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation*. 2020. arXiv: 2010.16061 [cs.LG].

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

[176] Alec Radford and Karthik Narasimhan. "Improving Language Understanding by Generative Pre-Training." In: 2018.

[177] Abdur Rahman, Abu Bakar Siddik Nayem, and Saeed Siddik. "Non-Functional Requirements Classification Using Machine Learning Algorithms." In: *International Journal of Intelligent Systems and Applications* (2023). URL: https://api.semanticscholar.org/CorpusID:259381940.

[178] K.M. Ashikur Rahman, Anwar Ghani, Rashid Ahmad, and Syed Haseeb Sajjad. "Hybrid Deep Learning Approach for Nonfunctional Software Requirements Classifications." In: *2023 International Conference on Communication, Computing and Digital Systems (C-CODE)* (2023), pp. 1–5.

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

[180] Kiramat Rahman, Anwar Ghani, Abdulrahman Alzahrani, Muhammad Usman Tariq, and Arif Ur Rahman. "Pre-Trained Model-Based NFR Classification: Overcoming Limited Data Challenges." In: *IEEE Access* 11 (2023), pp. 81787–81802.

[181] Mohamed Abdur Rahman, Md. Ariful Haque, Md. Nurul Ahad Tawhid, and Md. Saeed Siddik. "Classifying non-functional requirements using RNN variants for quality software development." In: *Proceedings of the 3rd ACM SIGSOFT International Workshop on Machine Learning Techniques for Software Quality Evaluation* (2019).

[182] Juan Enrique Ramos. "Using TF-IDF to Determine Word Relevance in Document Queries." In: *Computer Science* (2003). URL: https://api.semanticscholar.org/CorpusID:14638345.

[183] Abderahman Rashwan, Olga Ormandjieva, and René Witte. "Ontology-Based Classification of Non-functional Requirements in Software Specifications: A New Corpus and SVM-Based Classifier." In: *2013 IEEE 37th Annual Computer Software and Applications Conference* (2013), pp. 381–386.

[184] *ReFAIR: Toward a Context-Aware Recommender for Fairness Requirements Engineering*. Zenodo, 2024. DOI: 10.5281/zenodo.10470916. URL: https://doi.org/10.5281/zenodo.10470916.

[185] Jesse Read, Bernhard Pfahringer, Geoffrey Holmes, and Eibe Frank. "Classifier Chains: A Review and Perspectives." In: *Journal of Artificial Intelligence Research* 70 (2021), pp. 683–718. DOI: 10.1613/jair.1.12376. URL: https://doi.org/10.1613\%2Fjair.1.12376.

[186] Jörg Rech and Klaus-Dieter Althoff. "Artificial intelligence and software engineering: Status and future trends." In: *KI* 18.3 (2004), pp. 5–11.

[187] Brittany Reid, Markus Wagner, Marcelo d'Amorim, and Christoph Treude. "Software Engineering User Study Recruitment on Prolific: An Experience Report." In: *arXiv preprint arXiv:2201.05348* (2022).

[188] Maria Riaz, Jason Tyler King, John Slankas, and Laurie A. Williams. "Hidden in plain sight: Automatically identifying security requirements from natural language artifacts." In: *2014 IEEE 22nd International Requirements Engineering Conference (RE)* (2014), pp. 183–192.

[189] C. J. van Rijsbergen. "Information Retrieval." In: *ACM SIGSPATIAL International Workshop on Advances in Geographic Information Systems*. 1979.

[190] Abhishek Sainani, Preethu Rose Anish, Vivek Joshi, and Smita Ghaisas. "Extracting and Classifying Requirements from Software Engineering Contracts." In: *2020 IEEE 28th International Requirements Engineering Conference (RE)* (2020), pp. 147–157.

[191] Marco Saltarella, Giuseppe Desolda, and Rosa Lanzilotti. "Privacy Design Strategies and the GDPR: A Systematic Literature Review." In: *Interacción*. 2021.

[192] Steven L. Salzberg. "On comparing classifiers: Pitfalls to avoid and a recommended approach." English (US). In: *Data Mining and Knowledge Discovery* 1.3 (1997), pp. 317–328. ISSN: 1384-5810. DOI: 10.1023/A:1009752403260.

[193] M.A.F. Saroth, P.M.A.K. Wijerathne, and B.T.G.S. Kumara. "Automatic Multi-Class Non-Functional Software Requirements Classification Using Machine Learning Algorithms." In: *2024 International Research Conference on Smart Computing and Systems Engineering (SCSE)* 7 (2024), pp. 1–6. URL: https://api.semanticscholar.org/CorpusID:270398333.

[194] Kurt Schneider, Eric Knauss, Siv Hilde Houmb, Shareeful Islam, and Jan Jürjens. "Enhancing security requirements engineering by organizational learning." In: *Requirements Engineering* 17 (2012), pp. 35–56.

[195] Qais A. Shreda and Abualsoud Hanani. "Identifying Non-functional Requirements from Unconstrained Documents using Natural Language Processing and Machine Learning Approaches." In: *IEEE Access* (2024).

[196] P. Silva, C. Gonçalves, C. Godinho, N. Antunes, and M. Curado. "Using NLP and Machine Learning to Detect Data Privacy Violations." In: *Proceedings of IEEE Conference on Computer Communications Workshops*. 2020, pp. 972–977. DOI: 10.1109/INFOCOMWKSHPS50562.2020.9162683.

[197] Sukin Sim, Peter D. Johnson, and Alán Aspuru-Guzik. "Expressibility and Entangling Capability of Parameterized Quantum Circuits for Hybrid Quantum-Classical Algorithms." In: *Advanced Quantum Technologies* 2 (2019). URL: https://api.semanticscholar.org/CorpusID:166228228.

[198] J. Slankas and L. Williams. "Automated extraction of non-functional requirements in available documentation." In: *Proceedings of 1st International Workshop on Natural Language Anal-*

*ysis in Software Engineering (NaturaLiSE)*. 2013, pp. 9–16. DOI: 10.1109/NAturaLiSE.2013.6611715.

[199] John Slankas and Laurie A. Williams. "Access Control Policy Extraction from Unconstrained Natural Language Text." In: *2013 International Conference on Social Computing* (2013), pp. 435–440.

[200] John Slankas and Laurie A. Williams. "Automated extraction of non-functional requirements in available documentation." In: *2013 1st International Workshop on Natural Language Analysis in Software Engineering (NaturaLiSE)* (2013), pp. 9–16.

[201] Ian Sommerville. *Software Engineering*. 9th ed. Harlow, England: Addison-Wesley, 2010. ISBN: 978-0-13-703515-1.

[202] Ian Sommerville. "Software engineering." In: *America: Pearson Education Inc* (2011).

[203] Ian Sommerville and Pete Sawyer. *Requirements Engineering: A Good Practice Guide*. New York, NY, USA: Wiley, 1997.

[204] Ian Sommerville and Peter Sawyer. "Requirements Engineering: A Good Practice Guide." In: 1997.

[205] Ezekiel Soremekun, Mike Papadakis, Maxime Cordy, and Yves Le Traon. "Software fairness: An analysis and survey." In: *arXiv preprint arXiv:2205.08809* (2022).

[206] Ronnie de Souza Santos, Felipe Fronchetti, Savio Freire, and Rodrigo Spinola. *Software Fairness Debt*. 2024.

[207] James C. Spall. "Implementation of the simultaneous perturbation algorithm for stochastic optimization." In: *IEEE Transactions on Aerospace and Electronic Systems* 34 (1998), pp. 817–823. URL: https://api.semanticscholar.org/CorpusID:122669076.

[208] Jonas Stein, Ivo Christ, Nico Kraus, Maximilian Balthasar Mansky, Robert Müller, and Claudia Linnhoff-Popien. "Applying QNLP to Sentiment Analysis in Finance." In: *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)* 02 (2023), pp. 20–25. URL: https://api.semanticscholar.org/CorpusID:260125353.

[209] P. Szymański and T. Kajdanowicz. "A scikit-based Python environment for performing multi-label classification." In: *ArXiv e-prints* (2017). arXiv: 1702.01460 [cs.LG].

[210] Chuanqi Tao, Hongjing Guo, and Zhiqiu Huang. "Identifying security issues for mobile applications based on user review summarization." In: *Inf. Softw. Technol.* 122 (2020), p. 106290.

[211] Adi L Tarca, Vincent J Carey, Xue-wen Chen, Roberto Romero, and Sorin Drăghici. "Machine learning and its applications to biology." In: *PLoS computational biology* 3.6 (2007), e116.

[212] W. B. Tesfay, J. Serna, and K. Rannenberg. "PrivacyBot: Detecting Privacy Sensitive Information in Unstructured Texts." In: *Proceedings of Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*. 2019, pp. 53–60. DOI: 10.1109/SNAMS.2019.8931855.

[213] Lisa Torrey and Jude Shavlik. "Transfer learning." In: *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global, 2010, pp. 242–264.

[214] László Tóth and László Vidács. "Comparative Study of The Performance of Various Classifiers in Labeling Non-Functional Requirements." In: *Inf. Technol. Control.* 48 (2019), pp. 432–445.

[215] Grigorios Tsoumakas and Ioannis Katakis. "Multi-label classification: An overview." In: *International Conference on Data Mining and Knowledge Discovery*. Springer. 2007, pp. 1–15.

[216] V. Tzerpos and R.C. Holt. "MoJo: a distance metric for software clusterings." In: *Sixth Working Conference on Reverse Engineering (Cat. No.PR00303)*. 1999, pp. 187–193. DOI: 10.1109/WCRE.1999.806959.

[217] Vasily Varenov and Aydar Gabdrahmanov. "Security Requirements Classification into Groups Using NLP Transformers." In: *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)* (2021), pp. 444–450.

[218] Dusan Varis and Ondřej Bojar. "Sequence Length is a Domain: Length-based Overfitting in Transformer Models." In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Ed. by Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021, pp. 8246–8257.

[219] Asimina Vasalou, Alastair J Gill, Fadhila Mazanderani, Chrysanthi Papoutsi, and Adam Joinson. "Privacy dictionary: A new resource for the automated content analysis of privacy." In: *Journal of the American Society for Information Science and Technology (JASIST)* 62.11 (2011), pp. 2095–2105. ISSN: 1532-2882. DOI: 10.1002/asi.21610.

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

[221] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL].

[222] Sahil Verma and Julia Rubin. "Fairness Definitions Explained." In: *Proceedings of the International Workshop on Software Fairness*. FairWare '18. Gothenburg, Sweden: Association for Computing Machinery, 2018, 1–7. ISBN: 9781450357463.

[223] Andreas Vogelsang and Markus Borg. "Requirements engineering for machine learning: Perspectives from data scientists." In: *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*. IEEE. 2019, pp. 245–251.

[224] Gianmario Voria, Francesco Casillo, Carmine Gravino, Gemma Catolino, and Fabio Palomba. "RECOVER: Toward the Automatic Requirements Generation from Stakeholders' Conversations." In: *Under major revision in Transactions on Software Engineering* (2025).

[225] Wentao Wang, Kavya Reddy Mahakala, Arushi Gupta, Nesrin Hussein, and Yinglin Wang. "A linear classifier based approach for identifying security requirements in open source software development." In: *J. Ind. Inf. Integr.* 14 (2019), pp. 34–40.

[226] Ronald L. Wasserstein and Nicole A. Lazar. "The ASA Statement on p-Values: Context, Process, and Purpose." In: *The American Statistician* 70.2 (2016), pp. 129–133. DOI: 10.1080/00031305.2016.1154108.

[227] S.M. Weiss and C.A. Kulikowski. *Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*. San Mateo, CA: Morgan Kaufmann, 1991.

[228] Colin M. Werner. "Towards A Theory of Shared Understanding of Non-Functional Requirements in Continuous Software Engineering." In: *2022 IEEE/ACM 44th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)* (2022), pp. 300–304. URL: https://api.semanticscholar.org/CorpusID:244405874.

[229] Colin Werner, Ze Shi Li, Derek Lowlind, Omar Elazhary, Neil Ernst, and Daniela Damian. "Continuously Managing NFRs: Opportunities and Challenges in Practice." In: *IEEE Transactions on Software Engineering* 48.7 (2022), 2629–2642. ISSN: 2326-3881.

[230] Jules White, Sam Hays, Quchen Fu, Jesse Spencer-Smith, and Douglas C Schmidt. "ChatGPT Prompt Patterns for Improving Code Quality, Refactoring, Requirements Elicitation, and Software Design." In: *arXiv preprint arXiv:2303.07839* (2023).

[231] Dominic Widdows. "Geometry and Meaning." In: *Computational Linguistics* 32 (2004), pp. 155–158. URL: https://api.semanticscholar.org/CorpusID:17581.

[232] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in software engineering*. Springer Science and Business Media, 2012.

[233] David H. Wolpert. "Stacked generalization." In: *Neural Networks* 5.2 (1992), pp. 241–259. ISSN: 0893-6080. DOI: https://doi.org/10.1016/S0893-6080(05)80023-1. URL: https://www.sciencedirect.com/science/article/pii/S0893608005800231.

[234] Xusheng Xiao, Amit M. Paradkar, Suresh Thummalapenta, and Tao Xie. "Automated extraction of security policies from natural-language software documents." In: *SIGSOFT FSE*. 2012.

[235] G. Xu, C. Qi, H. Yu, S. Xu, C. Zhao, and J. Yuan. "Detecting Sensitive Information of Unstructured Text Using Convolutional Neural Network." In: *Proceedings of International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*. 2019, pp. 474–479. DOI: 10.1109/CyberC.2019.00087.

[236] Muhammad Younas, Dyg. Norhayati Abg. Jawawi, Imran Ghani, and Muhammad Arif Shah. "Extraction of non-functional requirement using semantic similarity distance." In: *Neural Computing and Applications* 32 (2019), pp. 7383–7397.

[237] Muhammad Younas, Karzan Wakil, Dayang Norhayati Abang Jawawi, Muhammad Arif Shah, and Ahmad Mustafa. "An Automated Approach for Identification of Non-Functional Requirements using Word2Vec Model." In: *International Journal of Advanced Computer Science and Applications* (2019).

[238] Jianlong Zhou and Fang Chen. *Human and Machine Learning*. Springer, 2018.

[239] Yaqin Zhou, Shangqing Liu, Jingkai Siow, Xiaoning Du, and Yang Liu. "Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks." In: *Advances in Neural Information Processing Systems*. 2019, pp. 10197–10207.

[240] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. "Large language models are human-level prompt engineers." In: *arXiv preprint arXiv:2211.01910* (2022).

[241] Zhi-Hua Zhou. *Ensemble Methods: Foundations and Algorithms*. 1st. Chapman & Hall/CRC, 2012. ISBN: 1439830037.

[242] Sebastian Zimmeck, Ziqi Wang, Lieyong Zou, Roger Iyengar, Bin Liu, Florian Schaub, Shomir Wilson, Norman M. Sadeh, Steven M. Bellovin, and Joel R. Reidenberg. "Automated Analysis of Privacy Requirements for Mobile Apps." In: *Network and Distributed System Security Symposium*. 2017.

# RINGRAZIAMENTI

*"Com'è cominciata io non saprei, la storia infinita con te..."*

Questa la canzone che mi è balzata in mente volendo ringraziare chi e cosa ha reso quest'avventura incredibile e indimenticabile.

Potrei partire dalla curiosità di comprendere come funzionasse il mondo della ricerca, quando, nell'estate 2021, chiesi agli advisors di tesi magistrale, Carmine e Vincenzo, di *"poter discutere su eventuali opportunità nella ricerca, anche per farmi un'idea di come e se continuare in quest'ambito"*. Ma era il periodo in cui stavano aiutandomi a rendere il lavoro di tesi una pubblicazione scientifica, quindi ero già immerso in questo mondo senza rendermene conto. Potrei partire dal lavoro di tesi magistrale, ma anche qui ci sono stati eventi e persone che han reso unica questa scelta. E potrei andare indietro all'infinito probabilmente, così come infinite sono le persone che vorrei ringraziare in queste poche righe. Per questo motivo andrò un po' a caso, così come a caso è iniziata la mia avventura all'Università di Salerno. Probabilmente è a quest'ultima che sono infinitamente grato, per avermi donato innumerevoli esperienze, difficoltà, ansie, paure, depressioni, gioie, amori, amici, soddisfazioni, opportunità e speranze. Grazie per esser stata fonte di vita di questi ultimi 10 anni, e grazie per aver dato da bere alla mia sete di conoscenza. Certo, ora non mi pare il caso di dilungarmi, ma son lieto di aver avuto l'occasione di essere tuo studente, così come lieto sono per gli advisors che mi hai assegnato, Carmine e Vincenzo.

Carmine e Vincenzo, io non solo sono onorato, ma vi ammiro per il coraggio e la pazienza che avete avuto nell'introdurmi e guidarmi nel mondo accademico. Chi ve l'ha fatto fare? Grazie infinite per la libertà concessami nello scegliere le diverse opzioni che si sono poste durante questi anni, ma soprattutto per l'umanità ed il supporto durante i momenti difficili, sia accademici che non. Mi avete fatto intuire cosa significhi essere professore universitario, o meglio, mi avete dato dato un'idea di cosa comporti, nonché della comprensione e della pazienza che bisogna avere con noi studenti.

Grazie per avermi invitato nella mia seconda dimora di questi ultimi 3 anni, il SeSa Lab. Ricordo i primi giorni di completo spaesamento, come a dire: "Ma io che ci faccio qui?". E poi le braciate, le parigine, i panini, le "mezze" giornate di lavoro. Quanta vita abbiam condiviso, quanto supporto mi hai donato e quanti punti di vista differenti mi hai offerto! Sinceramente non credo avrei potuto chiedere di meglio! È stato un onore ed un privilegio poter collaborare, e, se penso al gruppo iniziale ed al gruppo attuale, direi che sono due laboratori differenti! Son contento ed orgoglioso di questa crescita, spero di aver contribuito anche in minima parte, perché non puoi immaginare quanto bene e quante vicissitudini mi hai donato! Grazie a te ho conosciuto tantissime persone e fatto esperienze di vita uniche: partecipazione a workshops, eventi, congressi, conferenze, progetti nazionali e internazionali, periodo all'estero.

A proposito di quest'ultimo, voglio ringraziare il SEART group, per avermi accolto e ospitato nella fantastica città di Lugano. Dai paesaggi straordinari alle calorose persone con le quali mi son confrontato e confortato, e poi l'USI e il suo modus operandi, quanta precisione e quanta fantasia! Grazie per aver reso la mia prima esperienza lontano da casa unica e indimenticabile, mi auguro di esser stato di piacevole compagnia e di avervi lasciato un ricordo felice, perché quei giorni son stati per me infinitamente preziosi.

Anche se devo ammettere che la fortuna è sempre stata dalla mia parte. La fortuna di un padre che, con l'esempio e non con le parole, ti scolpisce nell'anima il valore dell'umiltà e dell'altruismo, che diventa una furia per una luce lasciata accesa o cinque minuti in più sotto la doccia, ma che riesce a rendere ogni strada meno pesante, ogni scelta meno spaventosa, perché sai che sarà lì, senza esitare un secondo, pronto a sostenerti, a capirti e, soprattutto, a sopportarti.

La fortuna di una madre che ti avvolge d'amore in ogni gesto, che si fa carico delle preoccupazioni di tutti senza mai tirarsi indietro, che lotta ogni giorno, spesso in silenzio, per vederti felice. Una donna capace di dimenticare se stessa pur di regalare un sorriso alla sua famiglia, un amore instancabile che non chiede nulla in cambio se non di saperti sereno.

Non ho trovato nulla al mondo che mi tocchi il cuore più di voi, quando crollate di sonno dopo giornate di sacrifici immensi. È in quei momenti che invidio tutta la vostra forza e tutta la vostra fragilità.

E poi mio fratello e mia sorella, dimostrazione che la vostra eredità più grande vive in loro: rivedo la stessa luce, la stessa tempra, la stessa caparbietà che ci ha cresciuti. Ecco cosa vedo in loro: due guerrieri, cadono e non si arrendono mai! E ogni volta che la vita li mette in ginocchio si rialzano più forti, più fieri, più vivi. Grazie, Armando e Stella, perché non immaginate quanta energia mi trasmettete, quanto mi consolate, quanto mi spronate: siete i miei complici silenziosi, la parte di me che conosce tutto senza dover parlare, le radici che camminano accanto a me.

Un infinito grazie va ai miei preziosi amici. Non vi ho mai raccontato tutto, forse non vi ho mai raccontato nemmeno un pizzico, ma è questo che vi rende speciali. Con voi non c'è bisogno di scavare nelle mie vicissitudini, vi basta uno sguardo, una battuta tagliente, un insulto ben piazzato, e mi fate capire che sapete esattamente chi sono. Siete quelli che mi prendono in giro senza pietà, che trasformano le fatiche in risate, e che, senza accorgermene, mi offrono sempre nuove prospettive. Con voi non devo spiegare: posso semplicemente essere. Sapere della vostra presenza, anche a distanza, nell'ironia tagliente e sgangherata, è una delle certezze più preziose che ho.

GRAZIE

A tutti quelli che non si son sentiti toccati da queste poche righe,
e a chi, purtroppo, non c'è più.

GRAZIE

A tutti, per esserci stati ed esserci proprio così come siete.
Questa tesi è vostra!

GRAZIE

Per permettermi di essere come sono.

Lascia un pensiero nella nostra tesi!

*La borsa di dottorato è stata cofinanziata con risorse del*

*Programma Operativo Nazionale Ricerca e Innovazione 2014-2020 (CCI 2014IT16M2OP005), risorse FSE REACT-EU,*

*Azione IV.4 "Dottorati e contratti di ricerca su tematiche dell'innovazione" e Azione IV.5 "Dottorati su tematiche Green"*

UNIONE EUROPEA
Fondo Sociale Europeo