Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Beyond domain dependency in security requirements identification

Francesco Casillo *, Vincenzo Deufemia, Carmine Gravino

Department of Computer Science, University of Salerno, Via Giovanni Paolo II, 132, Fisciano(SA), 84084, Italy

ARTICLE INFO

Keywords:
Security Requirements Classification
Natural Language Processing
Machine Learning
Transformers

ABSTRACT

Context: Early security requirements identification is crucial in software development, facilitating the integration of security measures into IT networks and reducing time and costs throughout software life-cycle.
Objectives: This paper addresses the limitations of existing methods that leverage Natural Language Processing (NLP) and machine learning techniques for detecting security requirements. These methods often fall short in capturing syntactic and semantic relationships, face challenges in adapting across domains, and rely heavily on extensive domain-specific data. In this paper we focus on identifying the most effective approaches for this task, highlighting both domain-specific and domain-independent strategies.
Method: Our methodology encompasses two primary streams of investigation. First, we explore shallow machine learning techniques, leveraging word embeddings. We test ensemble methods and grid search within and across domains, evaluating on three industrial datasets. Next, we develop several domain-independent models based on BERT, tailored to better detect security requirements by incorporating data on software weaknesses and vulnerabilities.
Results: Our findings reveal that ensemble and grid search methods prove effective in domain-specific and domain-independent experiments, respectively. However, our custom BERT models showcase domain independence and adaptability. Notably, the CweCveCodeBERT model excels in Precision and F1-score, outperforming existing approaches significantly. It improves F1-score by ~3% and Precision by ~14% over the best approach currently in the literature.
Conclusion: BERT-based models, especially with specialized pre-training, show promise for automating security requirement detection. This establishes a foundation for software engineering researchers and practitioners to utilize advanced NLP to improve security in early development phases, fostering the adoption of these state-of-the-art methods in real-world scenarios.

1. Introduction

Effective requirements gathering is essential for successful software development, as inaccuracies or omissions can disrupt the entire software lifecycle [1,2]. 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 [3–6].

The importance of security as an NFR is underscored by the substantial financial impact of security breaches [7], necessitating explicit identification and specification of security requirements, which are often implicitly distributed across requirement specifications [8,9].

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 [8–10], 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 [11].

To address domain-specific limitations, Munaiah et al. [12] 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.

* Corresponding author.
E-mail addresses: fcasillo@unisa.it (F. Casillo), deufemia@unisa.it (V. Deufemia), gravino@unisa.it (C. Gravino).

https://doi.org/10.1016/j.infsof.2025.107702
Received 10 January 2024; Received in revised form 27 November 2024; Accepted 24 February 2025

Recognizing these previous findings, in this paper 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 [13,14]. Firstly, we assess if shallow ML models can identify security requirements within the same domain. This motivates our first research question:

RQ1. How effectively can shallow machine learning algorithms, based on word embeddings, identify security-related requirements coming from the same domain?

A positive answer to RQ1 provides an alternative approach with respect to the methodologies previously implemented in [8–10], and avoiding the dependency on extensive datasets for model training as done in [10,11]. To address RQ1, 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) [8,9]. 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:

RQ2. 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 [11], which utilized 15 diverse projects, of which only three were from the industrial sector. We consider the work of Li and Chen [9] as the baseline for these comparisons, as it represents the best result achieved in this scenario.

To evaluate RQ2, 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. RQ2 exploration finds optimal hyperparameters for the top combination, and tests ensemble learning strategies as in RQ1. 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 then validate on requirements coming 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 [12], 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:

RQ3. How effective can pre-training context in BERT transformers be on the detection of security-related requirements?

To address RQ3 we pre-trained three distinct BERT models, each leveraging different datasets: (1) descriptions from CWE and the CVE; (2) the Defect Detection dataset [15] 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 pre-trained 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 paper 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 [16] is provided, enabling researchers to reproduce our findings or to further develop upon our work.

Organization of the paper. Section 2 presents background research on requirements classification and how security is typically analyzed in this context. Section 3 details the design of our empirical study for evaluating BERT-based models. Section 4 presents the quantitative results and discusses key findings. A discussion of threats to validity is included in Section 5, while Section 6 concludes the paper.

2. Related work

In this section, we provide an overview of existing approaches in the literature concerning the classification and detection of security requirements, contrasting them with our approach to highlight the advancements in our work.

While Varenov and Gabdrahmanov [17] made strides in classifying security requirements into predefined categories (Confidentiality, Integrity, Availability, Accountability, Operational, Access control, Other), exploiting a dataset containing solely security requirements and using BERT-based models, achieving an F1-score of about 78%, their focus was on multitasking classification rather than the identification task central to our study. We delve into the challenge of not just classifying but identifying security requirements, leveraging advanced BERT-based models for a more nuanced analysis.

Munaiah et al. [12] explored the domain-independent security requirement identification using one-class classification models. While notable, they primarily focused on using CWE database to build, by mean of TF-IDF [18], a vector space that served to train the One-SVM classifier, achieving an average F1-score of 67% when testing it on the industrial requirement specifications (ePurse, CPN and GP) that we used in our experimentation. Our approach, in contrast, aims to classify security requirements directly from requirement specifications as first attempt, offering a more direct and applicable solution to practitioners. In a second moment, we provide a solution to contrast with this approach, by giving a context to BERT models based on both code and CWE and CVE descriptions.

Wang et al.'s logistic regression model [19], which integrated project-related metrics to identify security requirements, introduced a novel perspective but faced limitations with a maximum F1-score of 60%. In contrast, our approach relies solely on requirement specifications, expanding its applicability to scenarios where design-related metrics are not accessible. The tool proposed by Riaz et al. [20] for identifying security-relevant sentences in natural language requirements documents, using a KNN classifier, achieved an F1-score of 80%. However, it required structured text documents, unlike our approach which can process requirements in their natural, unstructured form.

Cleland-Huang et al. [10] introduce an iterative method for training classifiers to identify non-functional requirements (NFR), tested through experiments with 15 requirement specifications from DePaul University MS students and a comprehensive document from Siemens Logistics and Automotive Organization. Although the classifier does not detect all NFRs, it supports analysts in identifying NFRs and expedites the analysis of large documents. Our approach concentrates solely on security requirements as NFRs, specifically avoiding poorly defined requirements from non-industrial domains.

Among similar studies, Knauss et al. [8] and Li and Chen [9] also performed both intra- and inter-domain experiments. Knauss et al. used a Bayesian classifier for automatic identification, while Li and Chen emphasized linguistic features and extended ontology. In our study, we capitalize on datasets from these investigations to perform intra- and inter-domain experiments, benchmarking results against their findings.

Finally, Mohamad et al. [11] explored the use of a classifier for security-related requirements, trained on 15 randomly selected, partially pre-labeled requirement documents totaling 3880 requirements. They conducted cross-project predictions where each specification served as a distinct group, similar to our RQ2, and evaluated the model against three United Nations regulations in the automotive sector, each varying in security relevance. While the findings demonstrate the viability of training a model across diverse domains, our focus is on refining techniques rather than relying on large data volumes. Consequently, we compare our results with their study in RQ3, highlighting differences in technique versus data quantity used in training.

3. 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.

3.1. Research questions

Our 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 fine-tuning for different NLP-based tasks, such as sequence classification. We start by assessing whether shallow ML models can be used for the specific task, which led to the formulation of the first research question:

RQ1 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 Fig. 1.

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., [21]).

TF-IDF [18] 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 [22, 23] 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 [24] 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 [25–27] 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 [13] is a transformer-based embedding model that can capture the bidirectional context of words. It is particularly effective in complex NLP 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 Fig. 1), our analyses encompass two aspects: the first facet of our investigation is to optimizing hyperparameters for the selected ML model (Step 2 in Fig. 1), and the second revolves around implementing ensemble learning techniques leveraging the top three combinations (Step 3 in Fig. 1). The pursuit of optimal hyperparameters is realized through Grid Search Cross-Validation (GridSearchCV) [28], 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 [29], bagging [30], and voting methodologies [31]. 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 3.2. These datasets have been chosen because previous studies exploited them [8,9], allowing us to carry out a comparison with existing classification models. 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 RQ1, 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. [8] and Li and Chen [9], 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 3.2 for more details).

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

RQ2 How effectively can shallow machine learning algorithms, based on word embeddings, identify security-related requirements coming from different domains?

To answer RQ2, 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

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

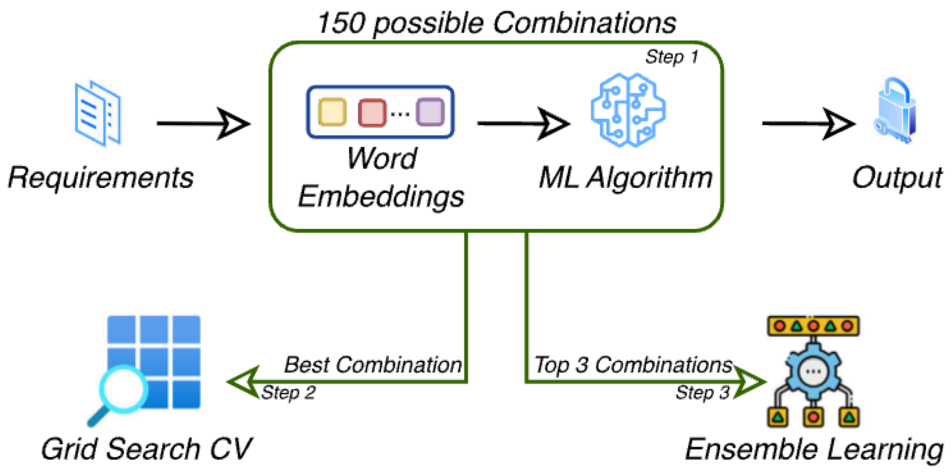

Fig. 1. Procedure employed to address RQ1 and RQ2.

trained on CPN & GPS are used to classify the requirements included in ePurse), thus performing a hold-out validation. See Section 3.2 for details regarding the complete list of training and test sets employed.

As in the case of RQ1, we conduct a deeper analysis to identify the optimal hyperparameters for the leading combination and practical experiments involving various ensemble learning strategies.

Thus, with RQ2, we want to evaluate the generalizability of the approaches considered in RQ1. For this reason, the training and test data must belong to different domains. However, obtaining training datasets for multiple 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 security 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 description of CWE-347,[3] which articulates that "*The product does not verify, or incorrectly verifies, the cryptographic signature for data.*" (example used also by Munaiah et al. [12]). 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:

RQ3 How effective can pre-training context in BERT transformers be on the detection of security-related requirements?

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

[3] https://cwe.mitre.org/data/definitions/347.html

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 [15], allowing it to learn and understand the nuanced language used to describe software weaknesses and vulnerabilities as well the related portions of code. See Section 3.2 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 underlying patterns and associations between security requirements and software vulnerabilities.

By employing this approach, as shown in Fig. 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 Fig. 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 2500 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 Fig. 2): Derived from the BERT base uncased model, we augment the pre-training phase by incorporating descriptions of CWE and CVE.
- **CodeBERT** ((3) in Fig. 2): CodeBERT [32] 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

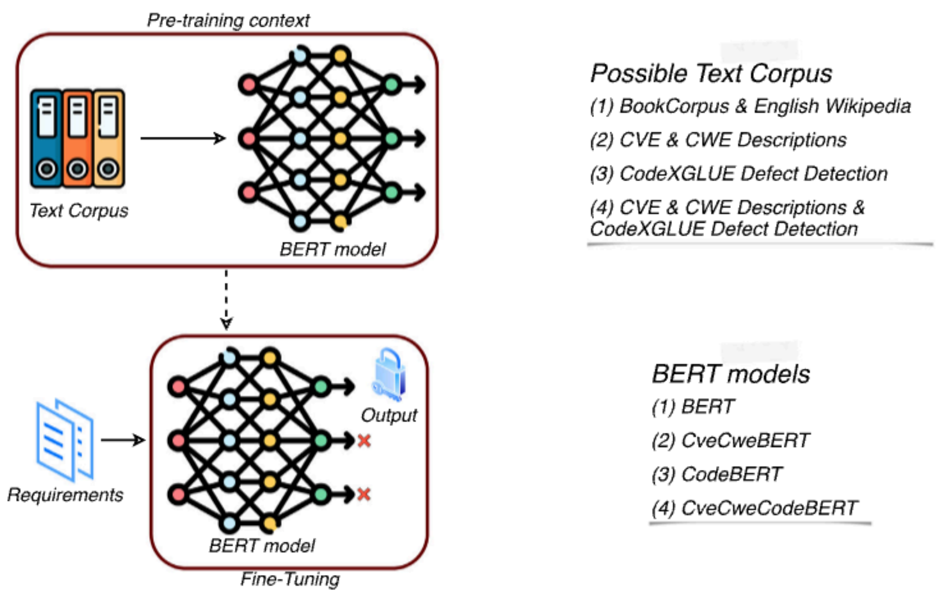

Fig. 2. Procedure employed to address RQ3.

Table 1
Requirements specifications considered in our empirical study.

Dataset (Abbreviation)	Total Requirements	Security Requirements	Avg. Words per Requirement
Common Electronic Purse (ePurse) [8,9,11]	124	83	29
Customer Premises Network (CPN) [8,9,11]	141	31	24
Global Platform Specifications (GPS) [8,9,11]	174	63	31
Collection of 12 projects (Train$_{Data}$) [11]	3369	652	25
United Nations regulations (Test$_{Data}$) [11]	362	47	43
PROMISE expanded version (Promise$_{exp}$) [33]	969	125	18

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 paper, we employ a version of CodeBERT fine-tuned on the CodeXGLUE Defect Detection dataset [15] for the Insecure Code Detection downstream task.

- **CweCveCodeBERT** ((4) in Fig. 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.

3.2. Datasets

Table 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 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. [8]; Munaiah et al. [12]; Li and Chen [9]; Mohamad et al. [11]) allowing us to perform a direct comparison with our proposal;
- 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 [34–36]. 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 [35], Knauss et al. [8], and Li and Chen [9], k = 10 is an appropriate value for this purpose. Indeed, 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 RQ2 and to assess the BERT-based models built to answer RQ3 because they require that the test and training data belong strictly to different domains.

Table 2
List of datasets used for training and test in each research question.

RQ1 10-fold cross-validation	RQ2 Training set (Test set)	RQ3 Different Pre-training setup (Fig. 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)	

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 final 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 RQ1 and RQ2, 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 2). The remaining datasets, i.e., Train$_{Data}$, Test$_{Data}$, and Promise$_{exp}$, are useful for evaluating the BERT-based approaches built to answer RQ3. 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[4] and CVE[5] descriptions catalogs. Below, we provide two examples of descriptions that help identify the displayed requirement from Promise$_{exp}$ as security-related:

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*".

3.3. Evaluation criteria

To assess the accuracy of our solutions, we employ widely used information retrieval criteria: Accuracy, Precision, Recall, and F1-score [37,38].

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 [8]. 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 [8,34]. 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 [39]. Specifically, we use McNemar's test to compare the performances of our prediction models [40,41]. Indeed, Salzberg [40], Japkowicz and Shah [41] 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 [40].

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 [42], 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.

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

Table 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

It is important to note that in our study, to answer RQ3 we compare three models,[6] 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) [43,44]. Since the number of tests is two, the correction applied is: 0.05/2 = 0.025 in the case of RQ3.

4. Results and discussion

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

4.1. RQ1 *How effectively can shallow machine learning algorithms, based on word embeddings, identify security-related requirements coming from the same domain?*

Table 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 RQ1. 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 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 terms of F1-score.

[6] Note that we implemented four BERT models but we "discarded" CweCve-BERT, 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.

Table 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

Table 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 results in Table 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.

4.1.1. *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

Table 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 [9].

Dataset	Stacking			Li and Chen [9]		
	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

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.

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 is 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.

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 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 as follow:

> 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 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 [9], 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.

4.2. **RQ2** *How effectively can shallow machine learning algorithms, based on word embeddings, identify security-related requirements coming from different domains?*

Table 7 displays the best results achieved by the 150 combinations of WE and ML models in the inter-domain experiment to address RQ2. 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 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

Table 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 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 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

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 RQ1), the designed combination still present interesting results, achieving 0.64 in terms of F1-score.

Results in Table 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.

4.2.1. 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. The FastText and GloVe embeddings show comparatively 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 more 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 security domain.

From Table 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 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,

Table 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 [9].

Dataset	GridSearchCV W2V & PAC			Li and Chen [9]		
	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

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 8 and 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 as follow:

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 10 compares the performance using GridSearchCV with Word2Vec and PAC (W2V & PAC) against the results obtained by Li and Chen [9]. W2V & PAC shows stronger performance in terms of Precision in most dataset combinations compared to Li et al.'s method.

Table 11
Results from different pre-training setups of BERT to detect security requirements.

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

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 more balanced trade-off between Precision and Recall across different dataset combinations, as indicated by its overall 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.'s method (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 [9].

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 most suitable approach for inter-domain identification of security-related requirements, despite the overall results favoring W2V & PAC.

4.3. ***RQ3*** *How effective can pre-training context in BERT transformers be on the detection of security-related requirements?*

Table 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 Test$_{Data}$ and the Promise$_{exp}$ dataset. The models evaluated are BERT, CweCveBERT, CodeBERT, and CweCveCodeBERT, already detailed in Section 3.1.

For BERT, testing on the Promise$_{exp}$ dataset yields similar results in terms of Precision, Recall, and F1-score compared to the results on Test$_{Data}$ dataset. In comparison, CweCveCodeBERT and CodeBERT exhibit strong performance when tested on Promise$_{exp}$, but lower on Test$_{Data}$. Table 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).

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 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.

Table 12
Inference performances of pre-trained BERT models on the industrial specifications.

Test set	BERT			CodeBERT			CweCveCodeBERT		
	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 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

4.3.1. Discussion

Our analysis reveal that standard BERT demonstrated robust performance, achieving the highest F1-score of 0.69 on Train$_{Data}$ (Test$_{Data}$) and maintaining a similar F1-score of 0.70 on 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 Test$_{Data}$, slightly improving to 0.62 on 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 Test$_{Data}$ (F1-score of 0.65) but excellent on Promise$_{exp}$ with an F1-score of 0.76.

This superior performance underlines the relevance of its pre-training context for the Promise$_{exp}$ dataset. CweCveCodeBERT, amalgamating features of both CWE/CVE and code-related data, shows balanced efficacy across Test$_{Data}$ and 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 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 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 RQ3 as follow:

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 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.

4.4. Implications for practitioners and researchers

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 RQ1, 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

Table 14
Results in terms of Precision, Recall, and F1-score achieved with *CweCveCodeBERT* compared with the results obtained by Mohamad et al. [11] and Munaiah et al. [12].

Dataset	CweCveCodeBERT			Mohamad et al. [11]			Munaiah et al. [12]		
	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

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, RQ2 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.

RQ3 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 RQ2. 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 toward 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. Threats to validity

This section elaborates on the threats to validity of the performed study, which stem mainly 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 [45]. 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`[7] method to measure Accuracy, the `precision_score`[8] method to measure Precision, the `recall_score`[9] method to measure Recall, and the `f1_score`[10] 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 [45]. In our study for exploring the applicability of our models to detect security-related aspects, we assume that these models are comparable to each other, as they come from the same libraries, i.e., Scikit-learn and Transformers.[11] 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 [45] 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.

[7] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html

[8] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html

[9] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html

[10] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

[11] https://huggingface.co/docs/transformers/main/en/index#transformers

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 considered 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 have made all the tools, scripts, and data available [16].

Conclusion Validity. It refers to the degree of reasonableness or correctness of conclusions regarding the defined null hypothesis and the performed statistical tests [45].

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 [42]. Thus, when null hypotheses are rejected, we consider the relationship between data and result to be reasonable.

6. Conclusion

This study embark on an exploratory journey to harness advanced ML techniques for the automatic detection of security-related requirements in software development. Through our comprehensive analysis, we have demonstrated the potential of both shallow ML and advanced BERT-based transformer models in addressing the nuanced challenges of security requirement identification. Indeed, our investigations across three distinct research questions revealed insightful findings. For RQ1, we established that ensemble learning methods, combining word-embedding techniques with shallow ML algorithms, are highly effective in identifying security requirements within the same domain. The results highlight that employing Word2Vec and combining NuSVC, Logistic Regression, and Ridge Classifier CV in a stacking ensemble allows us to achieve an F1-score of 0.82 and reached a maximum Precision of 0.84. For RQ2, we found that grid search techniques outperform other methods when identifying security requirements across different domains. In particular, fine-tuning the hyperparameters for the combination of W2V and PAC has proven to be the most effective strategy for identifying security requirements across different domains. This approach reached a maximum Precision of 0.73 and achieved a Recall and F1-score of 0.64 each.

In addressing RQ3, our investigation demonstrated that employing inter-domain BERT-based models, specifically CodeBERT and CweCve-CodeBERT, constitutes a significant advancement in the task. While both models showed promising outcomes, CweCveCodeBERT exhibited a significant improvement over the standard BERT model. As a result, CweCveCodeBERT was identified as the best model based on these findings. This model, pre-trained on code and CWE/CVE descriptions, demonstrated not only superior performance in terms of Precision and F1-score but also a remarkable adaptability across various industrial datasets, representing a significant step toward more reliable and efficient security requirement detection in software engineering. It addresses the critical gap in existing methodologies that predominantly rely on lexical and syntactic features, thus offering a more robust and nuanced approach to understanding and classifying security requirements. This means that the integration of models like CweCve-CodeBERT into standard software development pipelines could significantly enhance the early detection of security requirements, thereby improving the security posture of the developed software. Moreover, practitioners can explore customizing these models to their specific organizational contexts or project requirements, potentially enhancing their effectiveness further. Future research could delve into expanding the training context of models like CweCveCodeBERT, perhaps incorporating more diverse datasets or even data from specific high-risk. On the other hand, researchers can further explore the adaptability of these models across a wider range of domains, particularly focusing on their performance in less conventional or emerging fields. An essential area of future work lies in enhancing the interpretability of these models. Understanding how these models make their decisions is crucial for trust and widespread adoption.

In conclusion, this study not only contributes significantly to the existing body of knowledge in requirements engineering but also paves the way for practical applications that can reshape the approach to security in software development. By embracing the advancements in ML and NLP, both practitioners and researchers can continue to push the boundaries in the quest for more secure and robust software systems.

CRediT authorship contribution statement

Francesco Casillo: Writing – original draft, Writing – review & editing, Visualization, Validation, Resources, Project administration, Methodology, Investigation, Data curation, Conceptualization. **Vincenzo Deufemia:** Writing – original draft, Writing – review & editing, Validation, Supervision, Methodology. **Carmine Gravino:** Writing – original draft, Writing – review & editing, Validation, Supervision, Methodology.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

We have shared all data and scripts on Zenodo [16].

References

[1] I. Sommerville, P. Sawyer, Requirements Engineering: A Good Practice Guide, Wiley, New York, NY, USA, 1997.

[2] K. Pohl, Requirements Engineering: Fundamentals, Principles, and Techniques, first ed., Springer Publishing Company, Incorporated, 2010.

[3] F. Paetsch, A. Eberlein, F. 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, http://dx.doi.org/10.1109/ENABL.2003.1231428.

[4] Z. Kurtanović, 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, http://dx.doi.org/10.1109/RE.2017.82.

[5] Q.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, http://dx.doi.org/10.1109/MISE.2009.5069898.

[6] J. Slankas, L. Williams, Automated extraction of non-functional requirements in available documentation, in: Proceedings of 1st International Workshop on Natural Language Analysis in Software Engineering, NaturaLiSE, 2013, pp. 9–16, http://dx.doi.org/10.1109/NAturaLiSE.2013.6611715.

[7] P. Anthonysamy, A. Rashid, 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, http://dx.doi.org/10.1109/ICSE-SEIS.2017.3.

[8] E. Knauss, S. Houmb, K. Schneider, S. Islam, J. Jürjens, Supporting requirements engineers in recognising security issues, in: D. Berry, X. Franch (Eds.), Requirements Engineering: Foundation for Software Quality, Springer Berlin Heidelberg, 2011, pp. 4–18.

[9] T. Li, Z. Chen, An ontology-based learning approach for automatically classifying security requirements, J. Syst. Softw. 165 (2020) 110566, http://dx.doi.org/10.1016/j.jss.2020.110566.

[10] J. Cleland-Huang, R. Settimi, X. Zou, P. Solc, Automated classification of non-functional requirements, Requir. Eng. 12 (2) (2007) 103–120, http://dx.doi.org/10.1007/s00766-007-0045-1.

[11] M. Mohamad, J.-P. Steghöfer, A. Åström, R. Scandariato, Identifying security-related requirements in regulatory documents based on cross-project classification, in: PROMISE 2022, ACM, New York, NY, USA, 2022, pp. 82–91, http://dx.doi.org/10.1145/3558489.3559074.

[12] N. Munaiah, A. Meneely, P.K. Murukannaiah, A domain-independent model for identifying security requirements, in: 2017 IEEE 25th International Requirements Engineering Conference, RE, IEEE, 2017, pp. 506–511.

[13] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, 2019, arXiv:1810.04805.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, 2023, arXiv:1706.03762.

[15] Y. Zhou, S. Liu, J. Siow, X. Du, Y. 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.

[16] F. Casillo, V. Deufemia, C. Gravino, Beyond Domain Dependency in Security Requirements Identification, Zenodo, 2023, http://dx.doi.org/10.5281/zenodo.10438323.

[17] V. Varenov, A. Gabdrahmanov, Security requirements classification into groups using NLP transformers, in: 2021 IEEE 29th International Requirements Engineering Conference Workshops, REW, 2021, pp. 444–450, http://dx.doi.org/10.1109/REW53955.2021.9714713.

[18] J.E. Ramos, Using TF-IDF to determine word relevance in document queries, Comput. Sci. (2003) URL https://api.semanticscholar.org/CorpusID:14638345.

[19] W. Wang, N. Hussein, A. Gupta, Y. Wang, A regression model based approach for identifying security requirements in open source software development, in: 2017 IEEE 25th International Requirements Engineering Conference Workshops, REW, 2017, pp. 443–446, http://dx.doi.org/10.1109/REW.2017.56.

[20] M. Riaz, J. King, J. Slankas, L. 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, http://dx.doi.org/10.1109/RE.2014.6912260.

[21] C. Ferrara, F. Casillo, C. Gravino, A. De Lucia, F. 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, Association for Computing Machinery, New York, NY, USA, 2024, http://dx.doi.org/10.1145/3597503.3639185.

[22] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, 2013, arXiv:1301.3781.

[23] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, 2013, arXiv:1310.4546.

[24] J. Pennington, R. Socher, C.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.

[25] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching word vectors with subword information, 2017, arXiv:1607.04606.

[26] A. Joulin, E. Grave, P. Bojanowski, T. Mikolov, Bag of tricks for efficient text classification, 2016, arXiv:1607.01759.

[27] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, T. Mikolov, FastText.zip: Compressing text classification models, 2016, arXiv:1612.03651.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, É. Duchesnay, Scikit-learn: Machine learning in Python, J. Mach. Learn. Res. 12 (85) (2011) 2825–2830, URL http://jmlr.org/papers/v12/pedregosa11a.html.

[29] D.H. Wolpert, Stacked generalization, Neural Netw. 5 (2) (1992) 241–259, http://dx.doi.org/10.1016/S0893-6080(05)80023-1, URL https://www.sciencedirect.com/science/article/pii/S0893608005800231.

[30] L. Breiman, Bagging predictors, Mach. Learn. (1996).

[31] Z.-H. Zhou, Ensemble Methods: Foundations and Algorithms, first ed., Chapman & Hall/CRC, 2012.

[32] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang, M. Zhou, CodeBERT: A pre-trained model for programming and natural languages, 2020, CoRR abs/2002.08155 URL https://arxiv.org/abs/2002.08155.

[33] M. Lima, V. Valle, E. Costa, F. Lira, B. Gadelha, Software engineering repositories: Expanding the PROMISE database, in: Proceedings of the XXXIII Brazilian Symposium on Software Engineering, SBES '19, ACM, New York, NY, USA, 2019, pp. 427–436, http://dx.doi.org/10.1145/3350768.3350776.

[34] N. Ireson, F. Ciravegna, M.E. Califf, D. Freitag, N. Kushmerick, A. Lavelli, Evaluating machine learning for information extraction, in: ICML '05: Proceedings of the 22nd International Conference on Machine Learning, ICML '05, ACM, New York, NY, USA, 2005, pp. 345–352, http://dx.doi.org/10.1145/1102351.1102395.

[35] F. Chantree, Identifying nocuous ambiguities in natural language requirements, in: 14th IEEE International Requirements Engineering Conference, RE' 06, 2006, pp. 59–68.

[36] S. Weiss, C. Kulikowski, Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems, Morgan Kaufmann, San Mateo, CA, 1991.

[37] D.M.W. Powers, Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation, J. Mach. Learn. Technol. 2 (2011) 37–63.

[38] R. Baeza-Yates, B. Ribeiro-Neto, Modern information retrieval, 1999.

[39] R.L. Wasserstein, N.A. Lazar, The ASA statement on p-values: Context, process, and purpose, Amer. Statist. 70 (2) (2016) 129–133, http://dx.doi.org/10.1080/00031305.2016.1154108.

[40] S. Salzberg, On comparing classifiers: Pitfalls to avoid and a recommended approach, Data Min. Knowl. Discov. 1 (3) (1997) 317–328, http://dx.doi.org/10.1023/A:1009752403260.

[41] N. Japkowicz, M. Shah, Evaluating Learning Algorithms: A Classification Perspective, Cambridge University Press, 2011, http://dx.doi.org/10.1017/CBO9780511921803.

[42] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, A. Wesslén, Experimentation in Software Engineering, Springer Science & Business Media, 2012.

[43] J. Devore, N. Farnum, J. Doi, Applied Statistics for Engineers and Scientists, Cengage Learning, 2013, URL https://books.google.it/books?id=psg_CQAAQBAJ.

[44] W. Conover, Practical Nonparametric Statistics, Wiley series in probability and statistics, Wiley, 1999.

[45] D.T. Campbell, T.D. Cook, Quasi-Experimentation, RandMc-Nally, Chicago, IL, 1979.