Automated Identification of Security Requirements: A Machine Learning Approach

Armin Kobilica
armink@ieee.org
Information and Computer Science Department, KFUPM
Dhahran, Saudi Arabia

Mohammed Ayub
g201707490@kfupm.edu.sa
Information and Computer Science Department, KFUPM
Dhahran, Saudi Arabia

Jameleddine Hassine
jhassine@kfupm.edu.sa
Information and Computer Science Department, KFUPM
Dhahran, Saudi Arabia

ABSTRACT

Early characterization of security requirements supports system designers to integrate security aspects into early architectural design. However, distinguishing security related requirements from other functional and non-functional requirements can be tedious and error prone. To address this issue, machine learning techniques have proven to be successful in the identification of security requirements. In this paper, we have conducted an empirical study to evaluate the performance of 22 supervised machine learning classification algorithms and two deep learning approaches, in classifying security requirements, using the publicly availble *SecReq* dataset. More specifically, we focused on the robustness of these techniques with respect to the overhead of the pre-processing step. Results show that Long short-term memory (LSTM) network achieved the best accuracy (84%) among non-supervised algorithms, while Boosted Ensemble achieved the highest accuracy (80%), among supervised algorithms.

CCS CONCEPTS

• **Computing methodologies → Feature selection**; **Boosting**; **Classification and regression trees**; **Neural networks**.

KEYWORDS

Machine Learning, Security Requirements, Fast Pre-processing Techniques

ACM Reference Format:
Armin Kobilica, Mohammed Ayub, and Jameleddine Hassine. 2020. Automated Identification of Security Requirements: A Machine Learning Approach. In *Secure Software: Challenges, Opportunities and Lessons Learned, April 15-17, 2020, Trondheim, Norway.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3383219.3383288

1 INTRODUCTION

Requirements are generally expressed in natural language and suffer from several problems, such as incompleteness, inconsistency, redundancy, and vagueness [3]. These issues result from the fact that requirements are elicited from several stakeholders having different backgrounds and using different styles of communication. In addition, stakeholders may adopt heterogeneous requirements documentation practices, using different terminologies and specification standards. Such practices may produce requirements specifications, where functional and non-functional requirements are intertwined; hence, they are hard to detect and dissociate. This problem hinders, especially, the identification of non-functional requirements in an early stage of the development process, which is essential for the selection of proper software architecture and design.

In recent years, Security Requirements Engineering (SRE) has gained significant momentum from the requirements engineering community [11]. The early identification and integration of security requirements at the early stages of the development life cycle, would increase security awareness, mitigate potential threats, and reduce future security-related defects. Although, security belongs to a class of non-functional requirements (NFRs) related to system dependability, many security requirements are functional in nature [13]. However, the manual classification of security requirements is demanding, error-prone, and expensive (i.e., requires domain experts).

The last twenty years represent a blooming era for machine learning (ML) and its applications. Nowadays, ML-based techniques are well established in many disciplines. Machine learning approaches offer powerful tools to support various requirements engineering activities [3]. Several ML classifiers, e.g., Support Vector Machine (SVM), Random Forest (RF), Naive Bayes (NB), Logistic Regression (LR), Decision Tree (DT), and Neural Network (NN), were used in literature for the identification and classification of software requirements. Although, these classifiers have achieved, in general, remarkable results, they present different performance and accuracy levels depending on the context of use and data availability. Indeed, due to the vagueness/heterogeneous nature of requirements, automated ML tools may produce false positives and may lead to poor classification. In this paper, we conduct an empirical study to evaluate various ML-based automatic classification algorithms, using SecReq [4] dataset (enclosing more than 500 expert-labeled security requirements). The surveyed approaches employ different pre-processing and feature extraction techniques, which impacts the accuracy and performance of the proposed classifications. Furthermore, and to the best of our knowledge, this work is the first of its kind that (1) explores the use of Recurrent Neural Network (RNN)-based Long Short-Term Memory (LSTM) deep learning on SecReq, and (2) employs various k-nearest neighbors (KNN) and Ensemble-based ML classifiers on SecReq. In our study, we focus on the analysis of the fast pre-processing techniques.

EASE '20, April 15-17, 2020, Trondheim, Norway
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7731-7/20/04…$15.00
https://doi.org/10.1145/3383219.3383288

The remainder of this paper is organized into four sections. Related work is discussed in next section. The proposed empirical methodology is presented in Sect. 3. Experimental results are presented in Sect. 4 followed by a discussion in Sect. 5. Finally, Sect. 6 concludes the paper by summing up the main results and outlining future research directions.

2 RELATED WORK

In what follows, we present existing studies addressing (1) the classification of requirements into functional and non-functional, and (2) the classification of security requirements. A summary of the reviewed literature review is presented in Table 1.

2.1 Functional and Non-functional Requirements Classification

A large body of research is devoted to the study of structured and unstructured software requirements documents. More particularly, the automatic extraction and classification of requirements into functional (FR) and non-functional (NFR) [1, 6, 8, 14, 15, 18].

Kurtanović and Maalej [8] proposed a supervised machine learning approach for the classification of FRs and 4 subcategories of NFRs, namely, usability, security, operational, and performance requirements. The authors [8] used SVM on user comments, collected from Amazon software reviews. In addition, the approach employed various features, including, meta-data, lexical, syntactical, bag of words, bigrams and trigrams, filtering stop-words, punctuation, and POS. The authors claimed that POS tagging was among the most informative features. Different from the work presented in [8], Marinho et al. [10] studied the process of dataset generation using NFR framework catalogs, collected using lightweight systematic mapping. The study emphasized on the identification of various types of NFRs such as usability, security, and performance. Their experimental results have shown that security and performance classification scored higher precision and recall scores, i.e., ranging between 85% and 98%.

Khelifa et al. [6] proposed an SVM-based technique that (1) classifies a requirement change (user reviews expressed in natural language) as either Functional Change (FC) or Technical Change (TC), and (2) classifies NFR requirements (user reviews expressed in natural language) into 8 different classes, namely, functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability, and portability. In their experiment, the authors [6] used three datasets (two from Tera Promise (TP) and one from UCIrivine ML Repository) for a total of 1000 requirements (700 for training and 300 for testing) were used to conduct the experiments. Only 72 security requirements were considered. However, the authors failed to report their study results.

Alrumaih et al. [1] suggested that hybrid classification techniques can improve the classification accuracy. Their experimental work included a combination of NB, DT, and NN for automated requirements classification. In a closely related work, Winkler and Vogelsang [18] have shown that using NN-based automated tool improved classification accuracy by 11%.

Interestingly, the work of Winkler and Vogelsang [17] addresses the issue of capturing the explanation behind the generation of a specific output, by Neural Networks (NN) approaches. Their approach [17] traces back visually decisions made by neural networks, providing a feedback to help explain such decisions. The authors [17] built their prototype and tested it on 10000 samples. However, the authors admitted that their approach is only applicable to the network architecture presented and that it lacks generalization.

Mezghani et al. [12] proposed the use of k-means for detection of redundancies and inconsistencies in natural language requirements. The authors [12] used Part of Speech (POS) tagging for requirements pre-processing. The authors claimed that k-means provided very relevant results on industrial datasets, with the detection of redundancy scoring a higher precision, when combined with preprocessed data.

In a recent work, Tóth and Vidács [15] have conducted a comparative study of the performance of various classifiers in labeling non-functional requirements. A total of 12 classifiers were applied to Tera Promise (TP) NFR dataset, and 9 classifiers on the Stack Overflow dataset. The authors [15] reported that Multinomial NB, SVM, LR, and DT have notable performance, with Multinomial NB being the fastest in terms of execution time. It is also interesting to notice that the Fully Connected Network (FCN), constructed using Keras libraries and Tensorflow back-end, performed well with larger datasets. Although, 21 different classifiers were covered, this work overlooked fast pre-processing methods.

2.2 Security Requirements Classification

In order to develop secure and reliable software, security requirements must be analyzed with caution. Knauss et al. [7] studied the classification of security requirements using a tool based on a Bayesian Classifier (BC). Three types of security concerns were considered (1) security requirements (describes either the part of the system that should be secured or a property that may threaten security, if violated), (2) security-relevant (a coarse-grained or a potentially important requirement), and (3) security-related (detailed functional requirement). The proposed approach [7] can reliably identify most of the security-relevant requirements with a recall value of 0.9 and a precision value of 0.8. Furthermore, their tool can be used in combination with Heuristic Requirements Assistant (HeRA) and other elicitation tools, allowing for increased reusability and transferability.

Jindal et al. [5] performed security requirements classification using J48 decision tree in combination with text mining techniques to identify keywords related to the authentication-authorization (AA), access control (AC), encryption, and data integrity (DI). The experimental results showed that the model with DI achieved the best performance and it assisted analysts, who have no adequate knowledge on security.

Dekhtyar and Fong [2] proposed a security requirements identification model that uses TensorFlow convolutional neural networks and *Word2Vec* for the representation of words. As a baseline model, the authors used Naïve Bayes over word count and TF-IDF representations of requirements. Results show that *Word2Vec* embeddings of the words and CNN outperformed the base line model.

Security requirements are often mixed with other types of requirements. Many existing methods did not deliver the expected

Table 1: Related Work

Ref.	Classifiers	Dataset	Approach
[7]	BC	SecReq	Classification
[13]	k-NN, MNB, SMO	Private	Identification
[17]	NN	Private	Classification, Visualization
[5]	J48, DT	TP NFR	Classification
[8]	SVM, AdaBoost, Extra Tree, GBoost, RB	RE17 NFR, UC	Classification
[2]	NB and CNN	SecReq, TP NFR	Identification
[9]	NB, BN, LMT, J48, SMO, LDT, PART	SecReq	Identification
[15]	NB, SVM, LLR, DT, kNN, MLP	TP NFR, Stack Overflow	Comparison
[14]	NB, SVM, LLR, DT, kNN, MLP	TP NFR	Comparison
[6]	SVM	TP NFR, UCI	Classification
[1]	NB, DT, NN	NA	Review
[12]	K-means	Private	Detection
[18]	NN	Private	Comparison
[10]	Supervised ML with SGD	TP NFR	Dataset generation
[16]	LR	Private	Classification

efficiency due to their domain-dependency. In order to address this issue, Li [9] proposed a hybrid automation method for the security requirement identification which combines ontology and linguistic knowledge with ML techniques. Results from three experiments (with different settings) show that the proposed approach achieved notable performance over existing techniques and that it can be generalized for different application domains.

Furthermore, Riaz et al. [13] studied the identification of the security requirements using natural language artifacts with ML. The authors [13] proposed templates that are based on the security requirements context. For template extraction, clustering of sentences using k-medoids with Linear Discriminant Analysis (LDA) is applied with other techniques. Experimental results on the dataset (from six health care domains) revealed that the tool-assisted method obtained a precision value of 82% and a recall value of 79%.

Wang et al. [16], proposed a method for security requirements identification on open source projects using logistic regression (LR). Authors have used complexity and external resources as metrics. The proposed model is then compared with various flavors of LR and evaluated using a 10-fold strategy. Results proved that linear logistic classification has the highest precision, recall and F1-score in all projects.

Among all surveyed approaches, we notice that the following ML classifiers: SVM, NB, NN, RF are the most commonly used. Deep learning model is used relatively less frequently than other ML techniques. One possible reason behind this is the lack of availability of public large datasets.

3 PROPOSED APPROACH

Most of the presented approaches on automated classification of security requirements involve customized pre-processing linguistic and semantic techniques that aim to help extract proper features to better explain the main security annotations. These complex pre-processing techniques might represent an overhead and are mainly subjective. Our goal is to avoid such subjective pre-processing overhead and examine the performance of various machine learning algorithms when using fast prep-processing techniques.

More specifically, we propose an empirical study that invoves the use of two deep learning techniques (CNN and RNN-based) and twenty-two common ML classifiers (three Tree-based, LR, NB, six SVM-based, six KNN-based, five Ensemble-based) combined with two fast text pre-processing techniques, namely, word encoding and word embedding. The majority of ML classifiers are used in combination with word encoding, while only deep learning (CNN-based) network is combined with word embedding for the more appropriate text-to-image representation of the input. Our overall approach is depicted in Fig. 1.

3.1 SecReq Dataset

In order to evaluate the ML-based classification approaches, we have used the publicly available *SecReq* [4] dataset. *SecReq* [4], used in RE'17 data challenge competition, is composed of three industrial specifications: (1) Common Electronic Purse (ePurse) having 177 labeled entries, (2) Customer Premises Network (CPN) having 124 labeled entries, and (3) Global Platform Specification (GPS) having 210 labeled entries. The dataset has three labels (classes): *sec*, *nonsec*, and *unknown*. However, due to the low number of *unknown* samples, we only consider *sec* and *nonsec* categories for further pre-processing and classification.

3.2 Fast Pre-processing

we limit the pre-processing of data to the following two methods:

- **Word Encoding**: Word encoding techniques map the tokenized words of a sentence to numerical indices based on a certain vocabulary. In our case, we used complete SecReq dataset to create a vocabulary of unique words. These words are then saved into an array. This array acts as a base to document-to-sequence conversion of each example of dataset, allowing for a fast numerical representation required during the classification phase. We use word encoding for most of the studied ML algorithms.

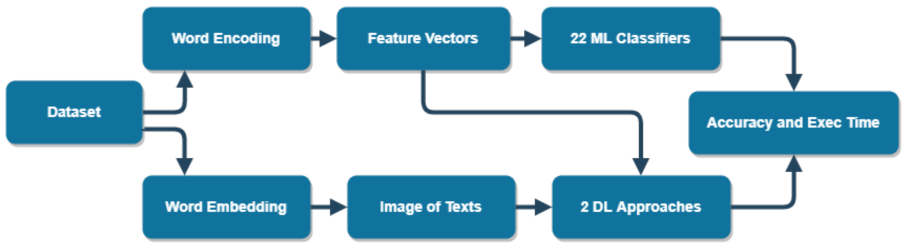


Figure 1: Proposed Approach

Table 2: ML Classification Results

Algorithm		Accuracy (%)	Training (sec)
Tree	Fine	73.30	14.243
	Medium	69.80	2.4946
	Coarse	61.00	1.283
	Logistic Regression	49.20	20.607
	Naïve Bayes	53.50	22.422
SVM	Linear	65.90	2.6987
	Quadratic	65.10	2.8048
	Cubic	70.00	2.08
	Fine Gaussian	71.00	1.5622
	Medium Gaussian	64.10	1.1934
	Coarse Guassian	63.30	1.024
KNN	Fine	72.90	1.346
	Medium	68.00	0.4981
	Coarse	63.30	0.5563
	Cosine	64.70	0.6042
	Cubic	67.30	1.7844
	Weighted	71.00	0.56449
Ensemble	**Boosted Trees**	**80.00**	**18.307**
	Bagged Trees	74.10	10.5
	Subspace Discriminant	64.50	7.6527
	Subspace KNN	**74.70**	**5.0231**
	RUSBoosted Trees	**77.30**	**9.5908**
LSTM		84.00	NA
CNN		80.00	NA

- **Word Encoding**: Word embedding is an improved version of the word encoding, that tries to capture semantics of words by using pre-trained word embedding model. We use word embedding and fixed length to create length-by-embedding-dimension image-like representation of the text, in order to accomodate the input of CNN deep learning network.

3.3 Supervised Machine Learning Classification

After applying fast pre-processing and in order to avoid overfitting and have better generalization of the results, we trained all 22 classifiers using 10-fold cross-validation. Table 2 shows the accuracy and execution times of all algorithms.

3.4 Deep Learning

In addition to the 22 ML classifiers, we experimented with the two most known deep learning techniques, RNN and CNN. LSTM (RNN-based) approach, with fast word encoding pre-processing achieved an accuracy of 84%, while CNN, in combination with word embedding and its image-alike representation did not exceed 80% in terms of accuracy. The results are presented in the last two rows of Table 2.

4 RESULTS ANALYSIS

Generally, all applied classifiers, except LR and NB, had at least one algorithm that achieved at least 70%. For example, in tree-based algorithms, the Fine approach achieved an accuracy of 73.3%, while SVM Fine Gaussian achieved an accuracy of 71%, Fine KNN achieved an accuracy of 72.9%, and the majority of Ensembles techniques achieved an accuracy above 70%. It is worth noting that results reported in literature achieved an accuracy of 95%, when complex linguistic and semantic pre-processing techniques are used. Such results may not be comparable to our results, since our goal is to examine to what extent shallow machine learning classifiers, with basic pre-processing technique, can achieve in terms of accuracy. While LSTM reached the highest accuracy level, its nature as a deep learning approach and its longer training time might not be suitable for instant and fast classification. Hence, in the context of security requirements, we would prefer the use of supervised machine learning classification approaches. In terms of accuracy, the ensemble-based classifiers have shown better performance comparing to the rest. Table 3 sorts the algorithms according to their achieved accuracy. When we exclude deep learning approaches, it is clear that ensemble-based algorithms have a better performance. Moreover, 9 out of the 22 approaches achieved an accuracy above 70%. This might give proper indication of what machine learning classifiers to be chosen when it comes to security requirements identification, regardless of the used pre-processing technique.

The accuracy ranking of Table 3 is reversed when we consider the execution time only. Table 4 illustrates the obtained results. It is worth noting that the classifiers achieving high accuracy require longer training times. The best accuracy score, achieved with the ensemble-based boosted trees, has one of the worst training time (18 seconds). It is interesting to point out that Weighted-KNN achieved an accuracy value of 71% within less than a second of training time and this might indicate that KNN-based classification should be considered first. It represents the best trade-off between accuracy and training time. This algorithm might also be very useful when employing more complex pre-processing techniques that increase overall accuracy. Surprisingly, LR and NB achieved the lowest accuracy levels and the worst training times. We may conclude that it would be more suitable to consider other ML approaches, when we deal with the classification of security requirements.

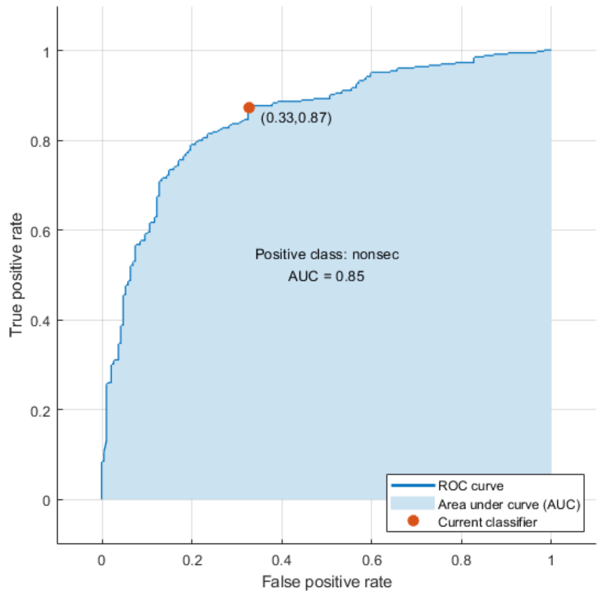
In addition to accuracy and training times, we were interested in the relationship between accuracy and the learning behavior.

Table 3: ML Classification - Sorted Accuracy Results

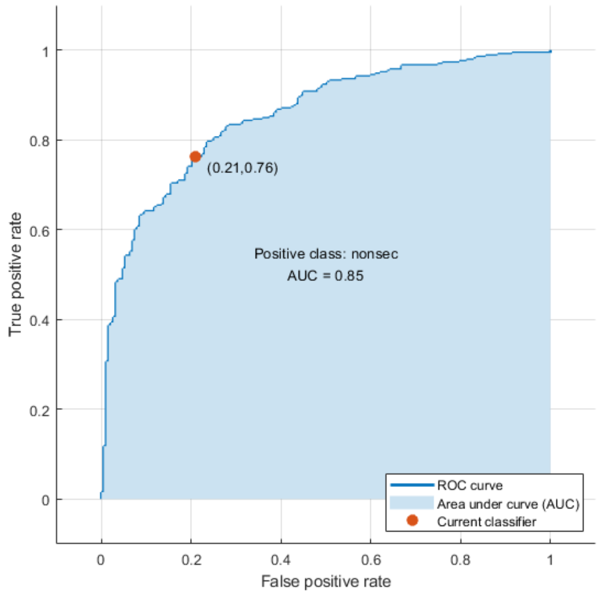
Algorithm		Accuracy (%)	Training (sec)
	LSTM	**84.00**	NA
Ensemble	**Boosted Trees**	**80.00**	18.307
	CNN	**80.00**	NA
Ensemble	RUSBoosted Trees	77.30	9.5908
Ensemble	Subspace KNN	74.70	5.0231
Ensemble	Bagged Trees	74.10	10.5
Tree	Fine	73.30	14.243
KNN	Fine	72.90	1.346
SVM	Fine Gaussian	71.00	1.5622
KNN	Weighted	71.00	0.56449
SVM	Cubic	70.00	2.08
Tree	Medium	69.80	2.4946
KNN	Medium	68.00	0.4981
KNN	Cubic	67.30	1.7844
SVM	Linear	65.90	2.6987
SVM	Quadratic	65.10	2.8048
KNN	Cosine	64.70	0.6042
Ensemble	Subspace Discriminant	64.50	7.6527
SVM	Medium Gaussian	64.10	1.1934
SVM	Coarse Guassian	63.30	1.024
KNN	Coarse	63.30	0.5563
Tree	Coarse	61.00	1.283
	Naïve Bayes	53.50	22.422
	Logistic Regression	49.20	20.607

Table 4: ML Classification - Sorted Execution Results

Algorithm		Training (sec)	Accuracy (%)
KNN	Medium	0.4981	68.00
KNN	Coarse	0.5563	63.30
KNN	Weighted	0.56449	71.00
KNN	Cosine	0.6042	64.70
SVM	Coarse Guassian	1.024	63.30
SVM	Medium Gaussian	1.1934	64.10
Tree	Coarse	1.283	61.00
KNN	Fine	1.346	72.90
SVM	Fine Gaussian	1.5622	71.00
KNN	Cubic	1.7844	67.30
SVM	Cubic	2.08	70.00
Tree	Medium	2.4946	69.80
SVM	Linear	2.6987	65.90
SVM	Quadratic	2.8048	65.10
Ensemble	Subspace KNN	5.0231	74.70
Ensemble	Subspace Discriminant	7.6527	64.50
Ensemble	RUSBoosted Trees	9.5908	77.30
Ensemble	Bagged Trees	10.5	74.10
Tree	Fine	14.243	73.30
Ensemble	**Boosted Trees**	**18.307**	**80.00**
	Logistic Regression	20.607	49.20
	Naïve Bayes	22.422	53.50
	LSTM	NA	**84.00**
	CNN	NA	**80.00**



(a) ROC Curve of the Highest Achieved Accuracy of Supervised Algorithm



(b) ROC Curve of the Second Highest Achieved Accuracy

Figure 2: The Best Results Achieved - ROC

To this end, we have chosen the two best approaches from supervised machine learning classifiers for further analysis. In Fig. 2, the receiver operating characteristic (ROC) curves denote that the automated approach using ML technique is promising, especially when we see that accuracy of the technique increased with more training times. On the other hand, Fig. 2 induces that *SecReq* dataset, with its 510 examples, might not be enough to generalize our findings and that the used algorithms did not reached their full potential in learning process.

It again proves common belief in ML that you cannot have enough data and more data gives more benefits. For further confirmation of the previous sentence, we applied PCA feature reduction techniques and re-run all experiments. We noticed that the performance of the classification deteriorated significantly, with none of the approaches achieved more than 74.9% in terms of accuracy. It is also interesting to emphasize that having enough data would help achieve higher accuracy without the need for complex pre-processing techniques. However, such claim cannot be verified until we have large corpus of labeled examples of security requirements that allows us to validate it.

5 DISCUSSION

Previous work on the automatic classification of security requirements in general, and *SecReq* dataset in particular, indicate that sophisticated and usually hard-coded pre-processing techniques are crucial in achieving any notable accuracy results in the overall machine learning process. While this might be true to the some extent, this impression conveys a message that secure software development must devote special attention to the pre-processing stage to capture accurately all security related requirements. Special attention involves exporting complex linguistic and semantic rules for each set of requirements to be able to perform automated identification. All other pre-processing techniques might not be as beneficial in the context of secure software development. In this work, we challenged this belief and with fast yet simple pre-processing techniques, we analyzed 24 machine learning classification algorithms and evaluated them in terms of accuracy and execution times. Promisingly, our results suggest that developers can highly rely on automated identification of security-related requirements without deeper analysis of the linguistic structure of the users' requirements.

6 CONCLUSION AND FUTURE WORK

Our empirical study showed the effectiveness of fast pre-processing as an initial filter for security-related and non-security-related requirements. Ensemble-based supervised machine learning classifiers (80% accuracy) and deep learning approaches (LSTM 84% and CNN 80%) have proven reliable with notable results. In all families of classifiers, the most effective were based on Fine technique and this may be confirmed in future evaluation studies.

Moreover, in our experiment using *SecReq* dataset, only 510 labeled entries were enough to achieve notable classification results. However, we strongly believe that more examples will positively affect accuracy results and having additional examples might prove to be sufficient to avoid complex pre-processing techniques. In addition, It would be interesting and worthwhile, if we could evaluate certain thresholds of the number of examples required for most machine learning classifiers to learn inner discriminative features without complex pre-processing techniques. To this end, more labeled data and an aggregation of the all available datasets related to security requirements, are needed. As part of our future work, we plan to help build such large datasets.

REFERENCES

[1] Hala Alrumaih, Abdulrahman Mirza, and Hessah Alsalamah. 2018. Toward Automated Software Requirements Classification. In *2018 21st Saudi Computer Society National Computer Conference (NCC)*. IEEE, New York, NY, USA, 1–6. https://doi.org/10.1109/NCG.2018.8593012

[2] Alex Dekhtyar and Vivian Fong. 2017. RE Data Challenge: Requirements Identification with Word2Vec and TensorFlow. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*. IEEE, New York, NY, USA, 484–489. https://doi.org/10.1109/RE.2017.26

[3] Henning Femmer, Daniel Méndez Fernández, Stefan Wagner, and Sebastian Eder. 2017. Rapid quality assurance with Requirements Smells. *Journal of Systems and Software* 123 (2017), 190 – 213. https://doi.org/10.1016/j.jss.2016.02.047

[4] Siv Hilde Houmb, Shareeful Islam, Eric Knauss, Jan Jürjens, and Kurt Schneider. 2010. SecReq dataset. http://www.se.uni-hannover.de/pages/en:projekte_re_secreq.

[5] Rajni Jindal, Ruchika Malhotra, and Abha Jain. 2016. Automated classification of security requirements. In *2016 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2016, Jaipur, India, September 21-24, 2016*. IEEE, New York, NY, USA, 2027–2033. https://doi.org/10.1109/ICACCI.2016.7732349

[6] Amani Khelifa, Mariem Haoues, and Asma Sellami. 2018. Towards a Software Requirements Change Classification using Support Vector Machine. In *Proceedings of the second Conference on Language Processing and Knowledge Management, Kerkennah (Sfax), Tunisia, October 17-18, 2018 (CEUR Workshop Proceedings)*, Vol. 2279. CEUR-WS.org, Aachen, 1–10. http://ceur-ws.org/Vol-2279/LPKM2018_paper_1.pdf

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

[8] Zijad Kurtanović and Walid Maalej. 2017. Automatically Classifying Functional and Non-functional Requirements Using Supervised Machine Learning. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*. IEEE, New York, NY, USA, 490–495. https://doi.org/10.1109/RE.2017.82

[9] Tong Li. 2017. Identifying Security Requirements Based on Linguistic Analysis and Machine Learning. In *2017 24th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, New York, NY, USA, 388–397. https://doi.org/10.1109/APSEC.2017.45

[10] Matheus Marinho, Danilo Arruda, Fernando Wanderley, and Anthony Lins. 2018. A Systematic Approach of Dataset Definition for a Supervised Machine Learning Using NFR Framework. In *2018 11th International Conference on the Quality of Information and Communications Technology (QUATIC)*. IEEE, New York, NY, USA, 110–118. https://doi.org/10.1109/QUATIC.2018.00024

[11] Daniel Mellado, Carlos Blanco, Luis E Sánchez, and Eduardo Fernández-Medina. 2010. A systematic review of security requirements engineering. *Computer Standards & Interfaces* 32, 4 (2010), 153 – 165. https://doi.org/10.1016/j.csi.2010.01.006

[12] Manel Mezghani, Juyeon Kang, and Florence Sèdes. 2018. Using k-Means for Redundancy and Inconsistency Detection: Application to Industrial Requirements. In *Natural Language Processing and Information Systems*, Max Silberztein, Faten Atigui, Elena Kornyshova, Elisabeth Métais, and Farid Meziane (Eds.). Springer International Publishing, Cham, 501–508.

[13] Maria Riaz, Jason King, John Slankas, and Laurie Williams. 2014. Hidden in plain sight: Automatically identifying security requirements from natural language artifacts. In *2014 IEEE 22nd International Requirements Engineering Conference (RE)*. IEEE, New York, NY, USA, 183–192. https://doi.org/10.1109/RE.2014.6912260

[14] László Tóth and László Vidács. 2018. Study of Various Classifiers for Identification and Classification of Non-functional Requirements. In *Computational Science and Its Applications – ICCSA 2018*, Osvaldo Gervasi, Beniamino Murgante, Sanjay Misra, Elena Stankova, Carmelo M. Torre, Ana Maria A.C. Rocha, David Taniar, Bernady O. Apduhan, Eufemia Tarantino, and Yeonseung Ryu (Eds.). Springer International Publishing, Cham, 492–503.

[15] László Tóth and László Vidács. 2019. Comparative Study of The Performance of Various Classifiers in Labeling Non-Functional Requirements. *Information Technology and Control* 48, 3 (2019), 432–445.

[16] Wentao Wang, Kavya Reddy Mahakala, Arushi Gupta, Nesrin Hussein, and Yinglin Wang. 2019. A linear classifier based approach for identifying security requirements in open source software development. *Journal of Industrial Information Integration* 14 (2019), 34 – 40. https://doi.org/10.1016/j.jii.2018.11.001

[17] Jonas Winkler and Andreas Vogelsang. 2016. *Towards Applicable Artificial Intelligence Tools in Requirements Engineering Using Visual Feedback*. Technical Report. Technische Universität Berlin, Berlin, Germany. https://doi.org/10.13140/RG.2.2.16725.83680

[18] Jonas Paul Winkler and Andreas Vogelsang. 2018. Using Tools to Assist Identification of Non-requirements in Requirements Specifications – A Controlled Experiment. In *Requirements Engineering: Foundation for Software Quality*, Erik Kamsties, Jennifer Horkoff, and Fabiano Dalpiaz (Eds.). Springer International Publishing, Cham, 57–71.