Security Requirements Classification into Groups Using NLP Transformers

Vasily Varenov
Innopolis University
v.varenov@innopolis.ru

Aydar Gabdrahmanov
Innopolis University
a.gabdrahmanov@innopolis.ru

2021 IEEE 29th International Requirements Engineering Conference Workshops (REW) | 978-1-6654-1898-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/REW53955.2021.9714713

***Abstract*—This study presents an implementation of sentence-level classification of security requirements into predefined groups. The method of this paper suggests using three models: BERT, XLNET, and DistilBERT for classification task and figures out evaluation metrics such as precision, recall, and F$_1$-score. We compiled a new dataset of 1086 security requirements of 7 classes collected from multiple existing datasets, such as PURE, SecReq and Riaz's dataset. The best-achieved result is DistilBERT's 78% F$_1$-score on the multiclass classification task. The main contribution of this study is the new multiclass dataset of security requirements and an example of how a deep transformer model can be used for requirements elicitation, which can be used as a basis for further improvement.**

I. Introduction

Software requirements are a very critical step in the software development process because as many studies show [1], the cost of mistakes during requirement analysis of development is much higher than on a later stage, and this is one of the most common reasons for failing a project [2]. Security requirements are especially important because security breaches cost millions of dollars each year, yet they are often neglected. They are non-functional requirements but they affect how the software system should operate and because of this they are often distributed all over requirements documentation, and often important details are missed.

The most typical way to express the software requirement is natural language. This approach is error-prone since it relies on the human ability to understand the language, for example, Firesmith [3] points out that many engineers confuse security requirements with countermeasures. In addition, customers omit many details in requirements specifications that are essential for developing software. That is why there exist many requirements specification approaches (eg. ISO/IEC/IEEE 29148[1]) that intend to alleviate problems by giving a set of prescribed categories for requirements and suggesting the details to be specified. Nevertheless, those standards are followed very loosely and there is still a problem of interpretation and classification of requirements.

Because of the aforementioned problems reading security requirements and extracting significant details for development can be really hard, and thus reducing the efficiency of product development. Automated detection and classification of security requirements can serve as the first step of requirements analysis and can help developers to find requirements relevant to their task fast and reliable.

Natural Language Processing (NLP) methods can be employed for a task of automated classification and detection. NLP is a subfield of computer science and linguistics which explores problems of automated processing and analysis of data presented in natural languages. NLP use various methods, most notable symbolic methods, statistical methods and neural networks. Symbolic methods are based on handcrafted rules by specialists and dictionaries. Statistical methods started the use of machine learning in NLP, they employed different statistical learning methods. Neural networks are the most modern method, it is characterized by the use of multilayered perceptrons together with machine learning. Currently, it is the best performing method and this study is going to focus on it.

Classification of sentences is one of the most common tasks in NLP. It is might be binary, e.g. is a sentence related to security or not, or might have multiple possible classes. In the latter case, there is a subdivision in multiclass and multi-label classification. In the first case, each sample belongs to one of the classes, in the latter case each sample might correspond to many labels or no labels at all. In this study, we focus on multiclass classification.

So, the research question is, can we develop an NLP model for automated classification of security requirements by security objectives with at least 80% F$_1$-score? 80% F$_1$-score is chosen as a goal because we consider a model with such performance shows that this approach is functioning and might be applicable in practice.

II. Literature Review

A. Related work

The software requirements are a significant part of software development, so the work on different systems to automate the requirements elicitation started a long time ago. Those systems cover different topics, such as finding similar requirements for reusing the existing requirements and corresponding solutions [4], classification of functional and non-functional requirements, classification of requirements of different categories, or binary classification of exactly one type of requirements, such as security requirements. The latter are going to be explored in more details due to their relevance to the classification of security requirements.

[1]https://standards.ieee.org/standard/29148-2018.html

Before the machine learning became widespread the solutions for these problems usually were based on some statistical methods using linguistic features, simple ones, like focusing on keywords in sentences [5], or more complicated ones, like part-of-speech tags, n-grams or checking if a sentence follows a given structure [6]. Those methods gave results, but those results were far from being useful to the real-world application since natural languages are heterogeneous and can not be fully expressed through limited set of handmade rules.

After the rise of interest in machine learning and deep learning appearance, many interesting and efficient different approaches in requirements elicitation were discovered. Winkler et al. [7] used convolutional neural networks for binary classification and Fong [8] compared different embeddings for convolutional neural networks and tried them on multiclass labelling task. She concluded that CNNs outperform Naïve Bayes model in most cases, getting 91.1% F_1-score in binary classification, and in multiclass labelling getting 77.2% average F_1-score, giving significant 27.5% boost in F_1-score. Kurtanović [9] compared different feature extractors and tried under- and oversampling strategies achieving maximum F1 score of 92% on F/NFR binary classifier with manual feature selection. Based on that model Dalpiaz et al. [10] made interpretable ML models by engineering a small set of human-understandable features without a significant drop in performance. Hey [11] employed modern deep learning model BERT and achieved F_1-scores of up to 94% on a binary task and 87% on a multiclass one.

Knauss et al. [12] set up a baseline in security classification by using Bayesing classifier and achieving a maximum of 84% F_1-score while achieving less then 60% whenever training and testing domains were different. Li [13] used ML models on top of on manually engineered linguistic features got 77% score on the same task but got an average F_1-score of 61% in across domain tasks, so his model gives more stable results to change of domain. Later [14] he attempted to improve his model by engineering more complex features based on ontology for security requirements and achieved 78% and 63% F_1-scores respectively. Munaiah et al. [15] tried a different approach by training one-class SVM on a dataset based on Common Weakness Enumeration without using negative examples and resulted with average 67.68% F_1-score on test data from a different domain.

B. Datasets

There are not many datasets on security requirements, and existing ones are quite small, relative to other NLP datasets, usually consisting of only a few hundreds of sentences, compared to other datasets having thousands and tens of thousands of sentences. Table I presents overview of found datasets, while this subsection describes them in more detail.

One of the most common datasets is the SecReq dataset, consisting of ePurse, CPN, and GPS datasets [12], which consist of around 500 requirement sentences, of which about 200 are security ones, these datasets are often used as a benchmark. Another common dataset is PROMISE, which includes 625 sentences, which of them assigned to one of 12 categories, 66 of them are security requirements, but the labelling is unbalanced and controversial, so there is relabeled version [10]. Slankas[16] compiled a dataset of health-related requirements with the same categories as PROMISE with 10,000 sentences, but for some reason, it was not used by anyone else after.

An important step of training NLP models is to finetune the language model on domain-specific data. A good dataset for this purpose is PURE[17] – 34,268 unlabeled sentences collected from 79 publicly available requirements. Another interesting dataset is Common Weakness Enumeration, which consists of a lot of texts related to requirements, and was used [15] to train one-class SVM classifier without negative samples. Also, CWE might be a good dataset for finetuning of language models, even though it was not used for this before.

Regarding datasets which have categories inside of security, there is only one multiclass dataset around 10,000 sentences big by Riaz et al. [18], which divide security requirements into Confidentiality, Integrity, Identification & Authentication, Availability, Accountability, Privacy. The main drawback of this dataset is that it is based on health-related requirements and was not tried by anyone else. Another possible approach is to use CWE-699 categorization of weaknesses[2], NIST SP 800-53 control families[3] or similar standards as a base for classes in a classification task, and use the description of weaknesses or controls as the dataset for training language model.

Dataset	Number of sentences	Description
PURE	34,268	Unlabeled requirements from 79 projects.
ePurse, CPN, GPS	510	Stadard dataset for security and non-security requirements classification.
PROMISE	625	14 classes labeled, including security class, compiled from 15 projects. Stadard dataset for FR/NFR requirements classification and multiclass classification.
Slankas	11,876	3568 of them are labeled with 14 classes, including security class, compiled from several medicine-related documents.
Riaz	10,963	5050 of them are security-relevant and labeled with 6 security objectives, compiled from several medicine-related documents.
NIST SP 800-53 controls	—	20 control families each of them consisting of several controls describing protective measures for systems, organizations, and individuals.
CWE-699 categorization of weaknesses	—	891 weaknesses with their description divided into 40 categories.

Table I: Comparison of different datasets

[2]https://cwe.mitre.org/data/definitions/699.html

[3]https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final

III. Methodology and Implementation

A. Metrics

In this study, we use three metrics for the evaluation of models performance – precision, recall, and F_1-score. These metrics are commonly used for machine learning classification tasks since they provide useful information on how models perform and which drawbacks they have. Here we will describe how they are calculated and which qualities they represent. Before that we need to define several variables:

- TP – true positives, or hit
- TN – true negatives, or correct rejection
- FP – false positives, or false alarm, type I error
- FN – false negative, or miss, type II error

$$precision = \frac{TP}{TP + FP}, \qquad recall = \frac{TP}{TP + FN}$$
$$F_1 = 2 \cdot \frac{precision * recall}{precision + recall}$$

1) Binary metrics: Precision is a metric that shows which proportion of our positively classified values were classified correctly.For example, precision would be equal to 1 if we classify only one sample positively and it would be a correct prediction, and it would be equal to 0 if we classify only one sample positively and it would be an incorrect prediction.

Recall is a metric that shows which proportion of our positive samples are correctly classified.For example, recall would be equal to 1 if we classify all samples as positives, since there would be no false negative conclusions, and it would be equal to 0 if we classify all samples as positives, since there would be no true positive conclusions.

F_1-score is a metric which is calculated as harmonic mean of precision and recall. F_1-score commonly used instead of simple arithmetic mean because it puts more emphasis on a lower metric, thus better showing if model is unbalanced in it's performance.

2) Multiclass metrics: Since all aforementioned metrics are for binary classification, some tweaks are required to make them work in multiclass classification case. There are many different ways to do it, and we are gonna use the most common and simple one. In this method, we treat the classification task as a multiple of binary classifications, one class versus the rest of the classes. Then we calculate metrics for each class and average them out. The other variation is to use a weighted average, so smaller classes will have a smaller impact on the final metric. But since our datasets are imbalanced and it is important to correctly classify all classes, and not just the most common one, we prefer an unweighted average.

3) Confusion matrix: Confusion matrices are useful for showing how a model performs in classification of different labels. Vertical axis true labels of samples, and horizontal axis shows predicted labels. So, if the sample is from class A and was misclassified as B, then on a confusion matrix it would increase count in cell (A, B). Confusion matrices are good for showing with which labels model confuses some label.

B. Preliminary experiments

For the preliminary experiments, binary classification on the SecReq dataset and multiclass classification were performed on the original Riaz's dataset. This greatly helped to a better understanding of datasets structure and model's training process and showed possible problems in future development. For ease of training DistilBERT model [19] from HuggingFace Transformers library[4] was chosen. Also we experimented with BERT [20] and XLNet [21] transformers, but they performed worse, while training process was more difficult because of their bigger sizes.

1) DistilBERT model: DistilBERT [19] is a transformer model based on BERT [20], but it has about half of the parameters of the original model and requires half of the processing power for training and on inference. Knowledge distillation technique [22] is what allowed such a decrease in complexity. For knowledge distillation teacher and student models are needed. The teacher is a more complex already trained model, while the student is an untrained more compact model with similar architecture. For training, the student special loss function is added. This loss corresponds to the internal activation of hidden layers of both models, such that the student is fitted not only to training data but also fitted to imitate the internal architecture of the teacher. In the instance of DistilBERT in result, performance drops in comparison with the teacher for only about 1%. For finetuning DistilBERT is trained the same way as regular BERT, with loss based on softmax function of predicted and actual labels.

One of the reasons why DistillBERT is a good model for our experiments is that we have a limited amount of train data and computing power, and bigger models require much bigger datasets and big computing clusters to perform well.

2) Results: Here will be presented only short description of results, for full description look at the Section 4. As can be seen, this model performed relatively well, comparing to the previous research. On SecReq [12] dataset it showed 87.17% F_1-score results from the start, while state-of-the-art NoRBERT model achieved up to 89% F_1-score on the similar binary classification task of security and non-security requirements [11]. On Riaz's dataset [18] finetuned DistilBERT multiclass classification achieved 54% F_1-score, which showed that training DistilBERT on multiclass classification task is relatively easy compared to other models, but after detailed inspection that Riaz's dataset were found to be not very clean.

C. Compiled Dataset

Because of the limitations of existing datasets, for this project, the new labelled dataset has to be made. A suitable basis for this is the PURE dataset – it consists of 79 documents from different fields with 34,268 unlabelled sentences. The drawback is that most of these documents are in PDF format and require a large amount of manual work to convert them to a more acceptable format for study, such as CSV or XLSX

[4]https://huggingface.co/transformers/

table. From extracted 5395 requirements, we've chosen only security ones, and ended up with 491 samples.

The next task was to labels security requirements for different classes. Because 491 requirements would not be enough for the fine-tuning model on the multiclass classification task, we decided to also use other datasets, SecReq and Riaz's, since these datasets had the most of security requirements. From SecReq were chosen only security requirements, which is 177 sentences. Riaz's dataset contains 5050 sentences labelled as related to security, but each sentence might have several security objectives and each objective can be a different level of security impact and had other quirks. After filtering and other data engineering, we've finished with 438 security requirements.

Collected 1086 security requirements were labelled with 7 possible labels: Confidentiality, Integrity, Availability, Accountability, Operational, Access control, Other. Labels distribution is shown in Table II. Initially, we used only the extended CIA triad – Confidentiality, Integrity, Availability, Accountability, but later realised that there were requirements that did not match with any of these security properties, while still being security requirements. Choice of the final labels was largely contributed by Silva and Danziger [23] and our experience with security requirements.

- Confidentiality – this property is satisfied when private data cannot be read by a person or application without appropriate access permission. A requirement labelled as confidentiality is expected to ensure that property.
- Integrity – this property is satisfied when private data cannot be written or deleted by a person or application without appropriate access permission. A requirement labelled as integrity is expected to ensure that property.
- Availability – this property is satisfied when the system functions as expected and users can perform their jobs. This includes protection from denial-of-service attacks, hardware and software faults, etc. A requirement labelled as availability is expected to ensure that property.
- Accountability (or Non-repudiation) – this property is satisfied when actions of users are recorded and users cannot alter the outcome of their actions or logs about these actions. A requirement labelled as accountability is expected to ensure that property
- Operational – this is a category of requirements that do not directly relate to the software product itself, but to the way it should be operated to remain secure – how to work with personnel, third parties, providers, legal requests, etc.
- Access control – this is a category of requirements that specify how authentication and authorization should be handled, while not being classified as confidentiality or integrity.
- Other – this is a category for requirements that is either too broad to be classified as one of the aforementioned categories, or too specific to be understood without a human with a deeper knowledge of the topic.

Class	Number of sentences
Confidentiality	389
Integrity	248
Availability	33
Accountability	130
Operational	112
Access control	29
Other	145
Sum	1086

Table II: Classes of composed dataset

D. *Main Experiments*

We've trained models for a maximum of 40 epochs and the best version showed 60% on all classes and 78% F_1-score on all classes except for the Other class. During different runs with different seeds results vary significantly, more than 10 per cent, so it was decided to implement 10-fold cross-validation [24], which would give results less dependent on random seeds set for data split and training.

1) 10-fold cross-validation: Cross-validation is one of the model validation technics which used to prove that model is generalizes well on different datasets and does not fall for selection bias. Essentially it is a way to prove that model would still perform well in practice and not on training data. 10-fold is a type of K-fold cross-validation, where K parameter is set to 10. The K-fold cross-validation process starts by splitting data into K equal disjoint subsamples. Then we train the model on all subsamples but one and validation performed on a left-out subsample. This is repeated so that each subsample once was used in validation and $K-1$ time used in training. After that, we should have K models and K results of the validation. Then all results are processed, for example, by averaging out them and using that result as a stable metric of the model's performance.

With 10-fold cross-validation model achieved 66.77% F_1-score. Also, cross-validation allowed us to build a confusion matrix using all dataset samples since all of them once were part of the validation subset.

Code and datasets can be found here: https://github.com/iambackend/thesis_code, https://github.com/gabdir/sec_req_classification.

IV. Results and Discussion

A. *Riaz's dataset*

For the multiclass classification with Riaz's dataset [18], only sentences with moderate or high security impact were chosen, and only the first label was used if a sentence had several of them. After training the DistilBERT model showed best result of 54% F_1-score, 58% precision and 53% recall on best iteration on the validation dataset. For this experiment train/validation split was 85/15. From the confusion matrices, it can be seen that the half of classes have only a few instances that were not mentioned in their publication, most likely they were omitted or merged with other classes for their experiments. The other half of classes is rather imbalanced, which can significantly decrease training results, and should be addressed in future work. Also, this dataset has a significant imbalance towards accountability class. The reason for this might be the fact that this dataset is composed of medical documents and accountability might be an important issue in that sphere.

B. Compiled dataset

For experiments with the compiled dataset, we also used DistilBERT model [19], since it performed well in initial experiments both in terms of performance and training complexity. During experiments, we trained the model on different subsets of classes, since some classes can affect because the classes representation in dataset and underlying requirements documents is highly variable and some classes significantly affect the final score of the model.

We've trained models for a maximum of 40 epochs and the best version showed 60% on all classes and 78% F_1-score on all classes except for the Other class.

Full results of all experiments are presented in Table III.

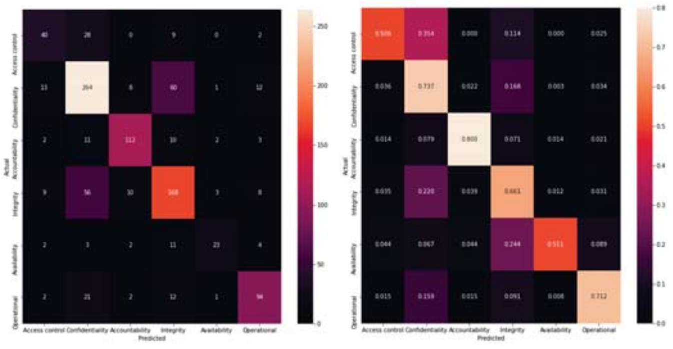

Figure 1: Confusion matrices of DistilBERT results on compiled dataset except Other class with 10-fold validation. Left figure is sample confusion matrix, and right one is normalized version.

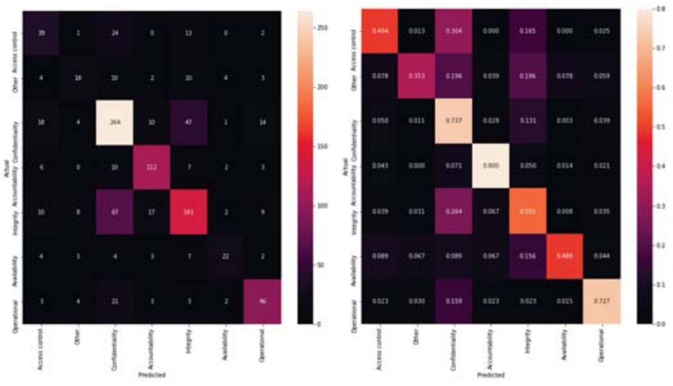

Figure 2: Confusion matrices of DistilBERT results on a full compiled dataset with 10-fold validation. Left figure is sample confusion matrix, and right one is normalized version.

C. Analysis of results

1) No Other class experiment: As it can be seen from confusion matrices for the experiment without Other class, there are several problematic classes. First of all, Confidentiality and Integrity, since these classes are closely related to each other: confidentiality ensures that data is not read by a non-authorized entity, and integrity ensures that data is not written by a non-authorized person. And often the difference is just in one word and hard to catch. Furthermore, there are plenty of requirements that do not specify which action is prohibited for non-authorized access. For example, "The System must allow the user to limit access to cases to specified users or user groups". Such requirements ensure both confidentiality and integrity qualities. Such requirements were labelled as Access Control, and requirements from this class are often misclassified as Confidentiality of Integrity.

Another problem is the Availability class. This quality should ensure that the system is working and available at all times. Firstly, this requirement has a relatively small number of samples – only 45 of 1086. This is because usually documentation does not go into details of availability requirements. And because this class shows bad performance and often gets misclassified. Especially Availability samples are often mistaken for Integrity samples. One of the things that availability ensures is that data is not lost to some unfortunate event and that the system has backups in case of unforeseen circumstances. Since it is related to writing the data and ensuring that it is not compromised, it is easy to confuse.

Lastly, Operational requirements are often confused with Confidentiality ones. This is because Operational requirements often emphasize how subcontractors and third-party products ensure various security requirements, and the most common of them is Confidentiality. Further, Operational requirements often cover the topic of personnel security, and it might be described similarly as requirements covering security of data, particularly Confidentiality requirements.

2) Full dataset experiment: As for the experiment on all classes, including the Other class, it does not give any particularly interesting insights. The Other class significantly decreases the final score (60% vs. 67% F_1-score). This class is very often misclassified, and only 35.3% of its requirements are recognised correctly. Requirements from this class the most often confused with Confidentiality and Integrity classes, but these are the most populous classes, so this can be explained as model defaults to using the most popular classes as a prediction.

From this, we can conclude that right now it is not sensible to train model including Other class. To this class belong the requirements which don't have enough reasons to belong to other classes, so they don't share any similarities other than not belonging to other classes. Thus, the ability of the model to classify Other requirements depends only on how good it is at the classification of other classes. And model would perform better on the classification of Other requirements only when there would be enough train data on other classes so the model would have a better understanding of their nature and would be able to tell that there are samples from neither of these classes.

Model	Dataset	Params	Precision	Recall	F_1-score
distilbert-base-uncased	SecReq	90/10 split	0.895	0.85	0.872
bert-base-uncased	Riaz	85/15 split	0.54	**0.55**	0.53
distilbert-base-uncased	Riaz	85/15 split	**0.58**	0.53	**0.54**
xlnet-base-uncased	Riaz	85/15 split	0.54	0.53	0.51
bert-base-uncased	Compiled, no Other	85/15 split	0.75	0.80	0.77
distilbert-base-uncased	Compiled, no Other	85/15 split	**0.80**	**0.82**	**0.78**
xlnet-base-uncased	Compiled, no Other	85/15 split	0.76	0.72	0.72
distilbert-base-uncased	Compiled, no Other	10-fold CV	0.710	0.665	0.668
distilbert-base-uncased	Compiled	10-fold CV	0.633	0.600	0.601

Table III: Results of different experiments on different datasets

D. Comparison with similar works

The main competitor to our work is Riaz's study because it is the only one we found which tried to classify classes inside of security requirements. The main difference between our studies is that their dataset was composed of medical documents, while our dataset included requirements from various sources. Also, they employed only classical machine learning algorithms for multilabel classification, such as Naïve Bayes, SMO and k-NN, while our work mainly focused on deep machine learning for multiclass classification.

Our results are not directly comparable because they used multilabel classification, and we used multiclass classification, and also we trained on different datasets, but there is no other benchmark for security requirements classification. We've got similar F_1-score to their work (78% comparing to their 80%). The main difference and advantage of our solution is the use of a pre-trained transformer deep learning model, which provides better generalization results [25] and requires a smaller dataset for fine-tuning.

V. Conclusion

The goal of this work was to explore new ways of security requirements elicitation, in particular, classification of them by means of deep learning models. For this, we used the pre-trained DistilBERT transformer model, which is a distilled version of the well-known BERT model and fine-tuned on already existing security requirements datasets. The model showed a good F_1-score on binary classification (87.7%) and mediocre results on part of Riaz's dataset (54%), partially because of the small, imbalanced dataset with disputable labelling. After that, we decided to create our own multiclass dataset, to alleviate problems with existing datasets. The model fine-tuned on this new data showed up to 78% F_1-score. These experiments showed that transformers are good for training even on small datasets. Also, experiments showed that our dataset is also not perfect and some improvement of it is possible.

There are several ways to improve on this work. The first one is to increase the size of the dataset. Right now it is a little over 1000 requirements with 7 defined classes, while usually deep learning models are training on datasets with thousands and tens of thousands of samples. So right now the dataset is too small to be considered fitting for a task. The use of a smaller, distilled model, such as DistilBERT, helps with this problem, since smaller models require less training data to converge, but only to a limited extent. Another problem is that the dataset has imbalanced classes. Partially this is because in the real world some classes of requirements are much less common than other ones, like Availability and Confidentiality, for example. Another reason is that our dataset currently is imbalanced in terms of origins of documents – half of the samples come from Riaz's dataset, which used only healthcare systems documentation, which has some specific terminology and preferences. For example, in medicine, it is important to track which medications were prescribed, and which doctor was responsible for which decisions. Therefore a large proportion of requirements belong to the Accountability.

Another important task is to improve the model's usability and adaptability. While DistilBERT and other similar models are good in understanding and classification, they are limited in many ways. First of all, they take simple text as input, while in reality usually requirements are passed in more complex formats, such as DOCX or PDF. These formats have a complex layout, requirements are spread in mixture with other information, there are no clear borders between different requirements and no standard format for documents. Also, there are plenty of human-introduced imperfections, such as typos, non-standard symbols or broken formatting. So converting regular software requirements documentation into requirements that are understandable by the model is the task on its own, and we see several possible ways to solve it. One way is that to make a special service for storing requirements, but the obvious drawback is it would be limiting requirements writers in their expression, and this service should be enforced inside of organization and old documentation should be imported to it. Another way is to create the instrument for extraction of requirements out of common text formats, either by coding it or by training a model for it. And the third way is to build a model which would take a document as an input instead of text.

Another problem is that the current model does not use the context of the requirements it is classifying. It does not use the nearby requirements as input and does not use the background information which might be given in other parts of a document, such as general product description or abbreviations explanations. Possibly a good technique for requirements elicitation would be a question answering model, which will read the document and after it will answer questions of developers. But this would require a different dataset with more complex labelling.

***Acknowledgements*—This work was conducted in the frame of the research project by Innopolis University and in relation to the VeriDevOps project funded by the European Commission under the grant H2020-EU.2.1.1. N957212.**

I thank Dr. Andrey Sadovykh, Kirill Yakovlev and the project team for supervision and help.

Bibliography cited

[1] L. Bormane, J. Gržibovska, S. Bērziša, and J. Grabis, "Impact of requirements elicitation processes on success of information system development projects," *Information Technology and Management Science*, vol. 19, Dec. 2016. DOI: 10.1515/itms-2016-0012.

[2] I. Attarzadeh and Siew Hock Ow, "Project management practices: Success versus failure," in *2008 International Symposium on Information Technology*, vol. 1, 2008, pp. 1–8. DOI: 10.1109/ITSIM.2008.4631634.

[3] D. G. Firesmith, "Engineering security requirements," English, *Journal of Object Technology*, vol. 2, no. 1, pp. 53–68, 2003, Cited By :161. [Online]. Available: www.scopus.com.

[4] J. L. Cybulski and K. Reed, "Requirements classification and reuse: Crossing domain boundaries," in *ICSR*, ser. Lecture Notes in Computer Science, vol. 1844, Springer, 2000, pp. 190–210.

[5] S. F. Tjong and D. M. Berry, "The design of SREE - A prototype potential ambiguity finder for requirements specifications and lessons learned," in *REFSQ*, ser. Lecture Notes in Computer Science, vol. 7830, Springer, 2013, pp. 80–95.

[6] A. Ferrari, G. Gori, B. Rosadini, I. Trotta, S. Bacherini, A. Fantechi, and S. Gnesi, "Detecting requirements defects with NLP patterns: An industrial experience in the railway domain," *Empir. Softw. Eng.*, vol. 23, no. 6, pp. 3684–3733, 2018.

[7] J. Winkler and A. Vogelsang, "Automatic classification of requirements based on convolutional neural networks," in *RE Workshops*, IEEE Computer Society, 2016, pp. 39–45.

[8] V. Fong, "Software requirements classification using word embeddings and convolutional neural networks," 2018.

[9] Z. Kurtanovic and W. Maalej, "Automatically classifying functional and non-functional requirements using supervised machine learning," in *RE*, IEEE Computer Society, 2017, pp. 490–495.

[10] F. Dalpiaz, D. Dell'Anna, F. B. Aydemir, and S. Çevikol, "Requirements classification with interpretable machine learning and dependency parsing," in *RE*, IEEE, 2019, pp. 142–152.

[11] T. Hey, J. Keim, A. Koziolek, and W. F. Tichy, "Norbert: Transfer learning for requirements classification," in *RE*, IEEE, 2020, pp. 169–179.

[12] E. Knauss, S. H. Houmb, K. Schneider, S. Islam, and J. Jürjens, "Supporting requirements engineers in recognising security issues," in *REFSQ*, ser. Lecture Notes in Computer Science, vol. 6606, Springer, 2011, pp. 4–18.

[13] T. Li, "Identifying security requirements based on linguistic analysis and machine learning," in *APSEC*, IEEE Computer Society, 2017, pp. 388–397.

[14] T. Li and Z. Chen, "An ontology-based learning approach for automatically classifying security requirements," *J. Syst. Softw.*, vol. 165, p. 110 566, 2020.

[15] N. Munaiah, A. Meneely, and P. K. Murukannaiah, "A domain-independent model for identifying security requirements," in *RE*, IEEE Computer Society, 2017, pp. 506–511.

[16] J. Slankas and L. Williams, "Automated extraction of non-functional requirements in available documentation," in *2013 1st International Workshop on Natural Language Analysis in Software Engineering (NaturaLiSE)*, 2013, pp. 9–16.

[17] A. Ferrari, G. O. Spagnolo, and S. Gnesi, "PURE: A dataset of public requirements documents," in *RE*, IEEE Computer Society, 2017, pp. 502–505.

[18] M. Riaz, J. T. King, J. Slankas, and L. A. Williams, "Hidden in plain sight: Automatically identifying security requirements from natural language artifacts," in *RE*, IEEE Computer Society, 2014, pp. 183–192.

[19] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter," *CoRR*, vol. abs/1910.01108, 2019. arXiv: 1910.01108. [Online]. Available: http://arxiv.org/abs/1910.01108.

[20] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018.

[21] Z. Yang, Z. Dai, Y. Yang, J. G. Carbonell, R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *CoRR*, vol. abs/1906.08237, 2019. arXiv: 1906.08237. [Online]. Available: http://arxiv.org/abs/1906.08237.

[22] G. Hinton, O. Vinyals, and J. Dean, *Distilling the knowledge in a neural network*, 2015. arXiv: 1503.02531 [stat.ML].

[23] M. A. da Silva and M. Danziger, "The importance of security requirements elicitation and how to do it," Project Management Institute, 2015.

[24] M. Ojala and G. C. Garriga, "Permutation tests for studying classifier performance," *J. Mach. Learn. Res.*, vol. 11, pp. 1833–1863, 2010.

[25] M. Baroni, "Linguistic generalization and compositionality in modern artificial neural networks," *Philosophical Transactions of the Royal Society B*, vol. 375, no. 1791, p. 20 190 307, 2020.