IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 11 September 2022, accepted 27 September 2022, date of publication 28 October 2022, date of current version 11 November 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3217752

RESEARCH ARTICLE

A Non-Exclusive Multi-Class Convolutional Neural Network for the Classification of Functional Requirements in AUTOSAR Software Requirement Specification Text

SANJANASRI JP[1], VIJAY KRISHNA MENON[2], SOMAN KP[1], AND ATUL K. R. OJHA[3,4]
[1]Center for Computational Engineering and Networking, Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Coimbatore 641112, India
[2]KeepFlying, Singapore 529482
[3]Data Science Institute, National University of Ireland Galway, Galway, H91 TK33 Ireland
[4]Panlingua Language Processing LLP, New Delhi 110028, India

Corresponding author: Sanjanasri Jp (jp_sanjanasri@cb.amrita.edu)

This work was supported in part by the Science Foundation Ireland (SFI) under Grant (SFI/12/RC/2289_P2 Insight_2), and in part by Panlingua Language Processing LLP.

ABSTRACT Software Requirement Specification (SRS) describes a software system to be developed that captures the functional, non-functional, and technical aspects of the stakeholder's requirements. Retrieval and extraction of software information from SRS are essential to the development of software product line (SPL). Albeit Natural Language Processing (NLP) techniques, such as information retrieval and standard machine learning, have been advocated in the recent past as a semi-automatic means of optimising requirements specifications, they have not been widely embraced. The complexity in the organization's information makes requirement analysis intricately a challenging task. The interdependence of subsystems and within an organisation drives this complexity. A plain multi-class classification framework may not address this issue. Hence, this paper propounds an automated non-exclusive approach for classification of functional requirements from SRS, using a deep learning framework. Specifically, Word2Vec and FastText word embeddings are utilised for document representation for training a convolutional neural network (CNN). The study was carried out by the compilation of manually categorised relevant enterprise data (AUTomotive Open System ARchitecture (AUTOSAR)), which were also employed for model training. Over a convolutional neural network, the impact of data trained with Word2Vec and FastText word embeddings from SRS documentation were compared to pre-trained word embeddings models, available online.

INDEX TERMS Functional requirement, software requirement specification, convolutional neural network, multi-layer perceptron, word embedding.

I. INTRODUCTION

Complex systems, such as automotive software systems, are organized into subsystems, designed, and built separately before being unified to accomplish the intended functionality. The aggregate of requirements documents intensifies as domain experts draft them for each subsystem. Comprehending the design concepts pertinent to the diverse needs challenges the requirements engineers, as cohesive information is dispersed among the multiple sets of system requirements. Requirements Engineering (RE) has evolved as an undeniable component of the software development process as a prospective workaround. RE is a sub-discipline of software engineering that deals with creating and refining software requirements specifications (SRS). The outcome of a collaborative software project hinges on its RE phase [1]. The system's complexity often consists of various interdependent inter-disciplinary modules, rendering SRS a challenging

The associate editor coordinating the review of this manuscript and approving it for publication was Chiu-W. Sham.

exercise. Hence, RE should double down on the systematic and repeatable approaches which verify that the system requirements are comprehensive, consistent, and relevant.

Software Requirement Classification (SRC) identifies the category to which a specific Software Requirement (SR) belongs [1]. There are two types of requirements in SRC. Functional Requirements (FRs) specify the user requirements and product features. In contrast, Non-Functional requirements (NFRs) describe a product's quality attributes, design and implementation constraints, and external interfaces such as security, reliability, and stability [2]. The steering system should turn the wheels towards the left when the driver turns left could be an example of the functional specification. The related non-functional requirement specifies how fast and smooth the wheels should have been aligned for the car to turn left. However, NFRs are beyond the scope of this study.

Under the functional requirements, there are separate subsystems/classes for each functionality. For instance, in the case of the chosen data set (AUTOSAR data), respective SRS documents are explicitly maintained for the Communication class and some generous requirements under the General category. The number and types of subsystems vary according to the enterprises.

In the automotive sector, developers of single subsystems of an automotive system are unaware of more than half of their dependencies on other subsystems [3]. For example, different teams develop the component subsystems comprising an automobile. These teams specify the requirements native to their subsystem but have minimal awareness of the relationships between other subsystems, and this results in ambiguity in data; the exact requirement is specified with different terms. Requirement engineers find it arduous to comprehend the relevant knowledge information dispersed across several constituent subsystems' conditions.

A. MOTIVATION

In the automotive industry, the software system's inputs, conditions, actions, and expected outputs are all comprehensive information in software requirements specifications. Despite the requirement documentation being well-described, automatic requirement categorization is challenging due to innate ambiguity in natural languages and the recourse to multiple terminologies and sentence patterns to represent a specific requirement [4]. SRC holds great potential as it classifies requirement statements that developers can comprehend while strategizing the system components pertinent to fulfilling a particular requirement. For example, effective classification makes prioritization and filtering of relevant requirements much more accessible.

The high ambiguity in the elicitation of requirement documents across the subsystems makes automatic classification more error-prone. For example, the Safety and Powertrain FR classes of AUTOSAR SRS document [5] share similar safety measures and associate software (SW) components. The classification turns more heinous if the dataset is imbalanced. In this paper, the selected AUTOSAR dataset also suffers from a severe imbalance among classes and ambiguity.

Therefore, the requirement classification carried out in this paper is designed as a simple neural network, instead viewed as a non-exclusive multi-class problem (NEMO). NEMO is a novel deep neural network (DNN) framework to perform multi-label classification introduced in this paper. NEMO turns this N-class classification problem into N two-class binary classification sub-problems. Each binary classifier is developed using the one-versus-the-rest approach. For example, among N binary classifiers for N classes, the classifier developed for class $c1$ is trained with the data from class $c1$ annotated as positive, and the data from all the other courses is annotated as unfavorable.

The outputs of the N-DNN models are unified to infer the classes of the requirements. This paper proposes a deep learning (DL) method for automatically classifying functional requirements from SRS documents.

B. OVERVIEW

This paper focuses on multi-class mappings of various functional specifications from AUTOSAR SRS documents. Data analysis techniques and a DL framework, specifically Convolutional Neural Network (CNN), classify functional software requirements into appropriate categories and make the classification process efficient and laborious. This paper investigates three specific aspects of artificial intelligence (AI) techniques for the RE of SRS documents.

- Word embeddings to represent the SRS document
- CNNs to build the classifier framework to classify functional requirements classes.
- The classifier network is intended to categorize documents in a non-exclusive way.

II. RELATED WORKS

The classification of software requirements with textual analysis is an evolving topic in software engineering research to enhance software quality. Casamayor et al. [6] proposed a semi-supervised recommender system model to aid requirement engineers in detecting and classifying NFRs from the descriptive text. This recommender system is built using the Expectation-Maximization method. Slankas and Williams [7] utilized various approaches, including K-Nearest Neighbours, Support Vector Machine, and Naive Bayes classifiers, to extract and classify NFRs from PROMISE NFR Datasets into 14 distinct categories and assessed their performance. Reference [8] classified requirements documents from the PROMISE repository into Functional Requirements (FR), NFRs, and subcategories of NFR s using the Support Vector Machines (SVM) method. Reference [9] employed a language model and most common keywords for identifying NFRs from requirements documents. In [10], Word2Vec embedding of the PROMISE dataset is sent through LSTM and GRU network as input for classification. Reference [11] utilized two text vectorization approaches and four machine learning methods to categorize the requirements into two categories (functional requirements and non-functional requirements). In [12], a multi-label requirement classifier based on CNN, classified NFRs into five categories: reliability, efficiency, portability, usability, and maintainability. Researchers have given less attention to FR work, referenced in fewer journals than NFR [13].

Reference [14] designed five integrated models for categorizing FR statements using Naive Bayes, Support Vector Machine (SVM), Decision Tree, Logistic Regression, and Support Vector Classification (SVC) algorithms to enhance their accuracy.

The novelty in this approach is that DL techniques automatically classify FRs of enterprise applications into appropriate categories. A binary classifier is designed for each class for non-exclusive classification, considering the innate dependencies and data imbalance among classes.

III. DATA DESCRIPTION

AUTOSAR SRS documents are available in Portable Document Format (PDF). These data are converted to raw text for further processing. The text is carefully pre-processed and cleansed, using appropriate regular expressions to remove noise at maximum. These documents contain various descriptions and details of some diagrammatic representations, which are not excluded in the analysis. The 21 functional requirements/specifications classes are selected from the AUTOSAR website to highlight data imbalance and interdependencies. Table 1 enlists the classes and their specifics. Figure 1 is a pie chart that shows visually how unbalanced the classes are in terms of the number of sentences.

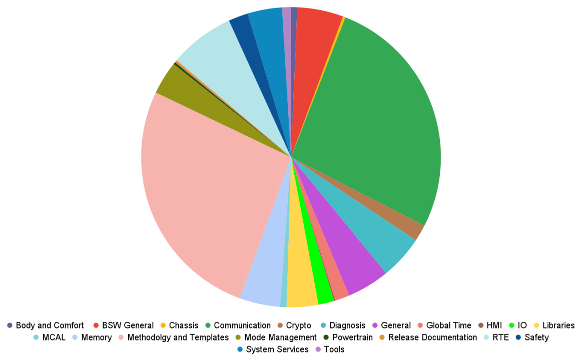

FIGURE 1. **Pie Chart for distribution of requirement classes on AUTOSAR SRS documentation (as in number of sentences).**

IV. CLASSIFICATION FRAMEWORK

Classification of software requirements involves four phases, as shown in Figure 2. They are

- **Data-pre-processing:** The raw unstructured data from the AUTOSAR document is converted to text format, cleaned, and normalized to ensure input data is eligible for further processing.
- **Vectorisation of texts:** The pre-processed data is given as an input to the AUTOSAR Doc2vec model to infer conceptual information from documentation as vectors (text embedding).
- **Classification:** The vectorized document from Phase 2 is used to train and predict the classification models used, Multi-Layer Perceptron (MLP) and CNN. The pre-processed document form each is given to separate 21 model classifiers for classification.
- **Evaluation:** The results of the requirement's label predictions and the true labels of these requirements are used to calculate the performance measures, presented in Section V.

A. PHASE 1: DATA PRE-PROCESSING

The AUTOSAR PDF files are converted into editable text documents prior to data cleansing. Tokens are produced from the raw text input during data pre-processing. Only the information that is comprehensible is retained after eliminating any other data such as tables, captions for figures and tables, page numbers, section titles, and punctuation. Sentences that are longer than 30 tokens are broken up into smaller ones. As most SW components utilise capital letters for abbreviation, they are excluded from case folding, which is used to unify the cases throughout the entire text. Figure 3 displays a line graph of the distribution of requirement classes in relation to the number of sentences(in thousands) on AUTOSAR SRS documentation before (in blue) and after pre-processing (red). When PDF files are converted to text format, the data is noticeably different since tables are formed as distinct lines during the conversion. As they don't offer information that is helpful for the categorization process, the table contents and figure names were all eliminated during the pre-processing.

B. PHASE 2: DOCUMENT REPRESENTATION

In order to carry out the classification that offers quantitative characteristics of the text, document representation/embedding is a crucial step, where the text is transformed into a vector representation that replicates semantic elements from documents. The smallest unit of a written or spoken language with a practical meaning is a word. The document representation is therefore constructed over the word embedding. Rich vector representations of words called word embeddings capture the syntactic and semantic links between words [15].

In this paper, the following two standard pre-trained word vector models are used: Word embedding for SRS documents is created using a) Word2Vec vector [16], and b) FastText's Common Crawl word vectors containing sub-word information. FastText and Word2Vec both aim to provide distinctive vector representations of words. FastText is a Word2Vec add-on [17]. Word2Vec and FastText both learn word vectors depending on their immediate surroundings. The manner of prediction varies between the two. Word2Vec predicts words using words, but FastText employs character n-grams to work at a finer level. To anticipate the words,

TABLE 1. Specification of AUTOSAR SRS documentation data.

Class Id	Requirement Classes	Before Data Pre-processing		After Data Pre-processing	
		No. of Lines	No.of Tokens	No. of Lines	No.of Tokens
0	Body and Comfort	1213	11798	705	11718
1	BSW General	9064	67987	5195	79244
2	Chassis	488	4628	284	4707
3	Communication	48500	389186	27736	365781
4	Crypto	3264	23418	1849	19884
5	Diagnosis	8616	54483	451	6171
6	General	8363	64597	4140	61260
7	Global Time	2875	20537	1353	17052
8	HMI	240	2009	149	2112
9	IO	3094	19574	502	7810
10	Libraries	6178	57052	568	8881
11	MCAL	1308	8006	193	3286
12	Memory	7962	52477	1009	13348
13	Methodology and Templates	48320	312867	6073	111606
14	Mode Management	6513	50580	3325	44839
15	Powertrain	452	3515	263	3305
16	Release Documentation	475	3530	353	3578
17	RTE	12804	81148	707	12096
18	Safety	3864	24663	408	6987
29	System Services	6776	47862	1256	22013
20	Tools	1748	13190	302	5431
	Total	182117	1313107	56821	811109

it makes use of character n-gram and subword information. FastText vectors containing sub-word information were chosen in particular for their capacity to deduce a superior semantic vector representation for unidentified words [18]. The word vectors pre-trained on millions of broad data may produce a poor semantic representation for certain words since the AUTOSAR dataset includes domain-specific terminology. Therefore, for improved semantic representation, the pre-trained word embedding is retrained using the cleaned AUTOSAR data.

Each class's pre-processed AUTOSAR data coalesce as a single text file. To create the sentence/document vectors, the merged huge text file is trained using the Doc2Vec method along with previously trained word vectors [19].

C. PHASE 3: DEEP LEARNING TECHNIQUES

The goal was to discover the multi-class mappings of different functional requirements to the 21 classes that are already known. For this investigation, basic deep neural network designs, an MLP, and a 1-D CNN were used. Then, the sentence embeddings were divided into training and testing groups, with training receiving the largest share (80%).

The classifier model is trained for all 21 classes separately with the one-against-all (OAA) strategy [20]. In OAA strategy, the text embedding for the one concerned class is labelled as '0', and the text embeddings of other classes are randomly sampled and labelled as '1' for (binary) classification. The sampling of data ensure that the deviation between the number of texts/sentences between class '0' and class '1' is kept minimal. For instance, the classifier model to be trained to detect whether the requirement belongs to the 'BSWGeneral' class, the text embeddings of the sentences belonging to the concerned class 'BSWGeneral' are labelled as '0', and the text embeddings belonging to the other 20 classes are randomly sampled, and labelled as '1'.

The one-against-all (OAA) technique is used to train the classifier model independently for each of the twenty-one classes. For (binary) classification in the OAA technique, the text embedding for the specific class is labeled as "0", while the text embeddings of other classes are randomly picked and labeled as "1". The sampling of data makes sure that there is little variation in the number of texts or phrases falling into classes "0" and "1." For example, the text embeddings of the sentences belonging to the class 'BSWGeneral' are labeled as '0' and the text embeddings belonging to the other 20 classes are randomly picked and labeled as '1' in the classifier model to be trained to determine whether the requirement belongs to that class. The data division ought to be roughly equal. For instance, if the class of interest only contains a little amount of data, such as 100 sentences (as in the case of the Human Machine Interface (HMI) in the AUTOSAR SRS document), then the text embedding from

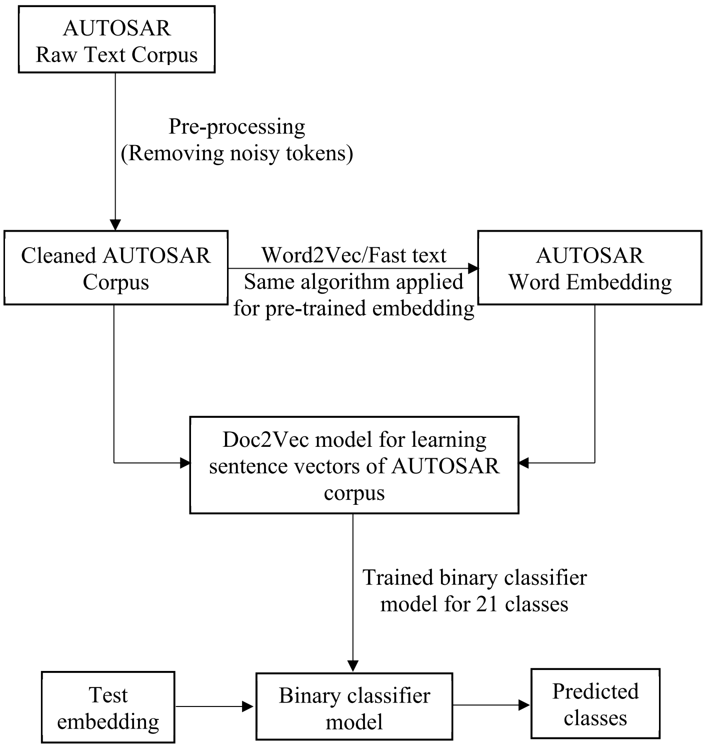
AUTOSAR Raw Text Corpus
Pre-processing (Removing noisy tokens)
Cleaned AUTOSAR Corpus
Word2Vec/Fast text Same algorithm applied for pre-trained embedding
AUTOSAR Word Embedding
Doc2Vec model for learning sentence vectors of AUTOSAR corpus
Trained binary classifier model for 21 classes
Test embedding
Binary classifier model
Predicted classes

FIGURE 2. Classification framework for functional requirements.

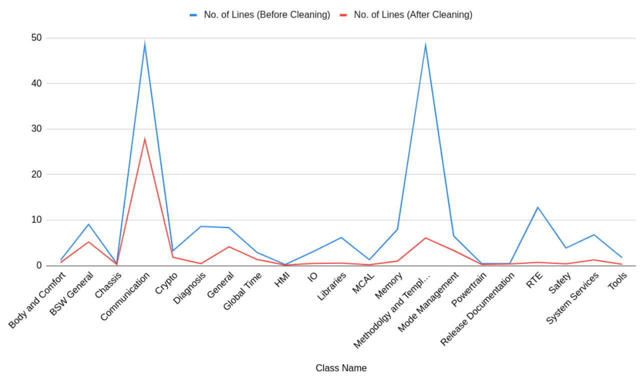

FIGURE 3. Distribution of requirement classes against the number of sentences (in thousands) on AUTOSAR SRS documentation before and after pre-processing.

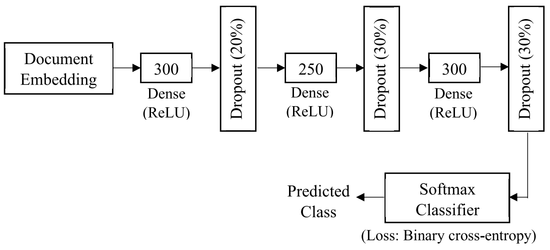

FIGURE 4. Architecture of MLP.

the remaining 20 classes is randomly selected, but only for an aggregate of 100 sentences.

Separate binary classifiers for each class help to prevent the over-fitting issue that is brought on by imbalanced classes. The best DL model classification parameters are found via grid search.

D. MULTI-LAYER PERCEPTRON

A subclass of feed-forward DNN is a multilayer perceptron (MLP). It uses one input layer and one output layer while having several hidden non-linear layers. In the proposed architecture, as seen in Figure 5, there are two hidden levels. According to mapping principles, the initial training data sets are matched with an appropriate targeted data set in this model.

Given a set of features $X = x_1, x_2, x_3 \ldots\ldots x_m$ and a target y,, where X is the sentence vector matrix, $x_i \in R^n$ is the n-dimensional vector corresponding to the $i^t h$ sentence in FR class and y is the $p \times 1$ vector of class labels. MLP uses Rectified Liner Unit (ReLU), a non-linear activation function and Binary Cross-Entropy as the loss function for classification. Figure 4 shows the architecture of the model.

E. ONE DIMENSIONAL-CONVOLUTIONAL NEURAL NETWORK

The Convolutional Neural Network (CNN) is a deep neural network that excels at automatically extracting features from input that may be processed in a "grid-like structure" [21]. In other words, CNNs are made to benefit from the location and order of the input pieces during learning, making them more suitable for pattern recognition tasks [22], [23]. Although MLP may learn non-linear models, its susceptibility to feature scaling is addressed with 1D-CNN. CNNs were first developed for image recognition, but they have also been successful in textual applications [24], [25].

Let $x_i \in R^n$ be the n-dimensional word vector corresponding to $i^t h$ sentence in FR class. A filter is used in a convolution process to create a new feature vector from a sentence vector (input). Here, $w \in R^k$ and $b \in R$ are the weight and bias terms, respectively and f is a non-linear activation function to learn the patterns in the vector. Rectified Liner Unit (ReLU) is the activation function and Binary Cross-Entropy is the loss function. Figure 5 depicts the model architecture.

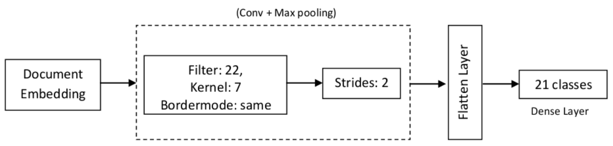

FIGURE 5. Convolution neural network framework for function requirement classification.

V. RESULTS AND DISCUSSION

Table 2 gives the overall performance of the DL networks stated above over two distinct feature scenarios: a retrained vector model and an original model using in-house vocabulary.

TABLE 2. Overall accuracy of MLP and CNN models with two data scenarios.

Experiment Specs	MLP		CNN	
	Loss	Accuracy(%)	Loss	Accuracy(%)
Word2Vec Original	0.98	71	1.00	72
Word2Vec Re-trained	0.93	73	0.98	75
FastText Original	0.92	73	0.93	75
FastText Retrained	0.91	75	0.84	77

Some FR classes that had data that was functionally equivalent in size to other classes' data (text content interpreted as functional requirements) were able to achieve model accuracy of greater than 70%. It seems sensible that the classes with the lowest model accuracy (20%) are also the ones that had the least amount of data. This suggests that, if there is balanced data available across classes, the problem has a very promising potential for success. Due to the highly skewed nature of the data and the higher model accuracy of the classes with more data, which may reach up to 95%, the total model accuracy in Table 2 is greater than 70%. Since the model's overall correctness cannot be assumed, it is vital to examine how each of the classes is performing using the suggested model. Figure 6 for the FastText Retrained Model shows the class-by-class accuracies for each of the twenty-one classes.

Small validation sets (less than 50 sentences) might cause statistical uncertainty since a single validation set could not accurately reflect the dataset as a whole. K-fold

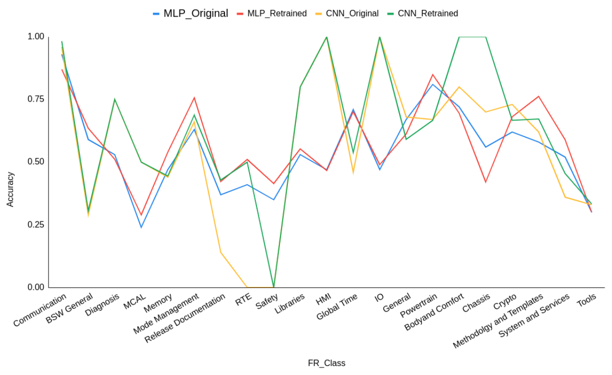

FIGURE 6. Accuracies of each binary classifier for the four DNN models trained by FastText pe-trained word embedding.

cross-validation is a method for resolving this conundrum. It involves repeating the training and testing process on k randomly chosen, non-overlapping divisions of the dataset and averaging the results over all k-folds [26].

Figure 7 shows that there are several classes with model accuracy as low as 0% or none at all. One can only assume that the explanation is a result of a lack of information. Additional infusions of data from these FR classes are required. There are still several FR classes with very little data (149 lines), such as HMI, yet with incredibly accurate models (98%) that exist. The apparent concern is how to increase trust in and confidence in the accuracy claims made by the 21 FR models. The accuracy of each of the 21 FR class models is shown in a histogram in Figure 7 along with a z-test statistic that indicates the level of confidence in each accuracy. The relative data size affects the z-test statistic (in number of lines, aka specifications). According to Figure 7, the accuracy varies depending on how general the FR classes are. The FR classes "Communication" and "BSW general," which have large amounts of data (more than 2000 lines), have strong model accuracy and a high level of confidence (<80%). The content of poorly performing classes like "HMI" and "Body and Comfort" is particularly ambiguous or overly generalized, which confuses the classifier model. As a result, these classes have strong model accuracy (<90%) in classification but low

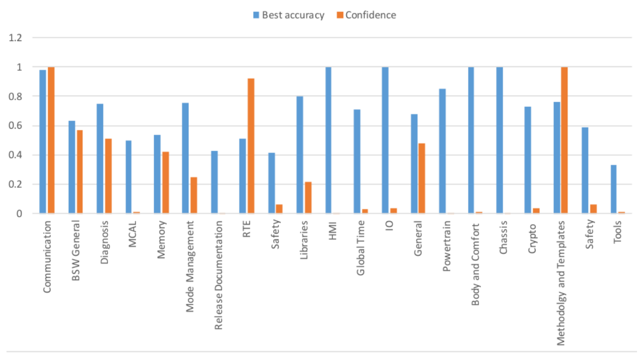

FIGURE 7. Best accuracy for each functional specification class and the its respective confidence statistic.

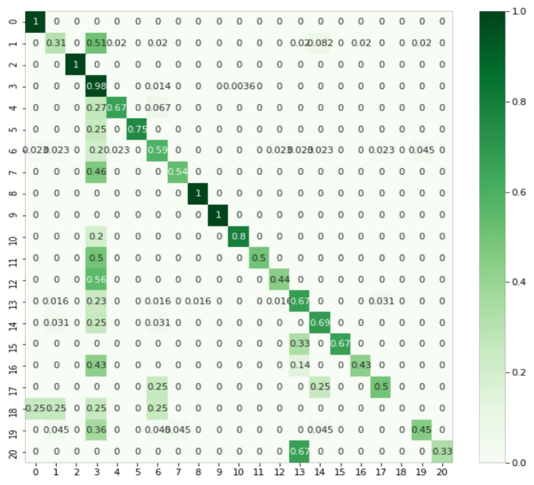

FIGURE 8. Confusion matrix generated from the CNN-Retrained-FastText model for 21 FR classes.

confidence scores. Figure 8, which displays a heat-mapped confusion matrix of 21 FR classes, makes this behavior obvious. The class id in Table 1 is aligned with the class id in the rows and columns of the confusion matrix. The columns display the forecasts, while the rows represent the actual facts. Figure 8 makes it evident that several of the low confidence classes in the confusion matrix predict false positives relative to the more confident classes. For instance, of all the courses, Class '3' ('Communication') is by far the most precise and certain of all the classes, as seen in Figure 7 in contrast, poorly performing classes like class '8' ('HMI'), '20' ('Tools'), and '1' ('Body and Comfort') have a large number of false positives due to a large amount of data that these FR classes have access to

Finally, we assert that despite the multi-class classifier network's widespread use in multi-label classification, it might not be a workable option in cases when data are few. With more balanced data, it will undoubtedly become better, but training efforts will be well behind. For the model to achieve an accuracy and confidence level of above 90%, deeper and more complex architectures are required. From a computational point of view, this will be a training challenge. The suggested classifier model considers the possibility that FR requirements might simultaneously define several classes.

VI. CONCLUSION

The general viability of explainable artificial intelligence models during automotive requirements engineering is discussed in the article. According to research, Model-Based Engineering is widely employed in the automobile industry and is even used by certain practitioners for requirements engineering. Domain-specific terms, however, have caused a number of issues with requirements engineering methods at automotive businesses. In this study, we present an NLP pipeline for categorizing functional requirements from AUTOSAR SRS papers into several types. MLP and CNN were used in the classification model's development. All 21 classes of AUTOSAR documents were used to train the twenty-one binary classifier model for multi-label categorization. Along with the word vectors of the sentences that are inferred from the pre-trained word vectors, Word2Vec and FastText, the Doc2Vec model vectorizes the sentences in AUTOSAR documents. The word vectors were produced using pre-trained online embedding and re-training the current embedding model using internal data. In both cases, retrained vector classifier models outperformed an initial vector model in terms of accuracy. The maximum accuracy, however, is provided by the retrained vector CNN classifier model (77%).

REFERENCES

[1] M. Lima, V. Valle, E. Costa, F. Lira, and B. Gadelha, "Software engineering repositories: Expanding the PROMISE database," in *Proc. 33rd Brazilian Symp. Softw. Eng.*, Sep. 2019, pp. 427–436.

[2] E. D. Canedo and B. C. Mendes, "Software requirements classification using machine learning algorithms," *Entropy*, vol. 22, no. 9, p. 1057, Sep. 2020.

[3] A. Schlutter and A. Vogelsang, "Knowledge representation of requirements documents using natural language processing," in *Proc. NPL4RE (REFSQ Workshop)*, Utrecht, The Netherlands, Mar. 2018.

[4] A. Rashwan, O. Ormandjieva, and R. Witte, "Ontology-based classification of non-functional requirements in software specifications: A new corpus and SVM-based classifier," in *Proc. IEEE 37th Annu. Comput. Softw. Appl. Conf.*, Jul. 2013, pp. 381–386.

[5] (Jun. 2021). *AutoSAR Documnetation*. [Online]. Available: https://www.autosar.org/nc/document-search/

[6] A. Casamayor, D. Godoy, and M. Campo, "Semi-supervised classification of non-functional requirements: An empirical analysis," *INTELIGENCIA Artif.*, vol. 13, no. 44, pp. 35–45, Jan. 2010.

[7] J. Slankas and L. Williams, "Automated extraction of non-functional requirements in available documentation," in *Proc. 1st Int. Workshop Natural Lang. Anal. Softw. Eng. (NaturaLiSE)*, May 2013, pp. 9–16.

[8] Z. Kurtanovic and W. Maalej, "Automatically classifying functional and non-functional requirements using supervised machine learning," in *Proc. IEEE 25th Int. Requirements Eng. Conf. (RE)*, Sep. 2017, pp. 490–495.

[9] M. Younas, K. Wakil, D. Jawawi, M. A. Shah, and A. Mustafa, "An automated approach for identification of non-functional requirements using Word2Vec model," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, pp. 539–547, Jan. 2019.

[10] A. Rahman, N. A. Tawhid, and S. Siddik, "Classifying non-functional requirements using RNN variants for quality software development," in *Proc. 3rd ACM SIGSOFT Int. Workshop Mach. Learn. Techn. Softw. Qual. Eval. (MaLTeSQuE)*, Tallinn, Estonia, Aug. 2019, pp. 25–30.

[11] E. D. Canedo and B. C. Mendes, "Software requirements classification using machine learning algorithms," *Entropy*, vol. 22, no. 9, p. 1057, Sep. 2020.

[12] M. Sabir, C. Chrysoulas, and E. Banissi, *Multi-Label Classifier to Deal With Misclassification Non-Functional Requirements*. Cham, Switzerland: Springer, May 2020, pp. 486–493.

[13] T. Tamai and T. Anzai, "Quality requirements analysis with machine learning," in *Proc. ENASE*, Setubal, Portugal, 2018, pp. 241–248.

[14] N. Rahimi, F. Eassa, and L. Elrefaei, "An ensemble machine learning technique for functional requirement classification," *Symmetry*, vol. 12, no. 10, p. 1601, Sep. 2020.

[15] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proc. 1st Int. Conf. Learn. Represent. (ICLR)*, Scottsdale, AZ, USA, May 2013, pp. 1–12.

[16] (Jun. 2021). *Google News Pre-Trained Vectors*. [Online]. Available: https://code.google.com/archive/p/word2vec/

[17] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Trans. Assoc. Comput. Linguistics*, vol. 5, pp. 135–146, Jun. 2016.

[18] S. Jp, V. K. Menon, S. Kp, S. Rajendran, and A. Wolk, "Generation of cross-lingual word vectors for low-resourced languages using deep learning and topological metrics in a data-efficient way," *Electronics*, vol. 10, no. 12, pp. 1–23, 2021.

[19] V. T. Priyanga, J. P. Sanjanasri, V. K. Menon, E. A. Gopalakrishnan, and K. P. Soman, "Exploring fake news identification using word and sentence embeddings," *J. Intell. Fuzzy Syst.*, vol. 41, no. 5, pp. 5441–5448, Nov. 2021.

[20] S. Kp, R. Loganathan, and A. Vadakkepatt, *Machine Learning With SVM and Other Kernel Methods*. India: PHI Learning, 2009.

[21] A. Conneau, H. Schwenk, L. Barrault, and Y. LeCun, "Very deep convolutional networks for text classification," in *Proc. 15th Conf. Eur. Chapter Assoc. Comput. Linguistics (EACL)*, Valencia, Spain, vol. 1, Apr. 2017, pp. 1107–1116.

[22] A. Jacovi, O. S. Shalom, and Y. Goldberg, "Understanding convolutional neural networks for text classification," in *Proc. EMNLP Workshop BlackboxNLP, Analyzing Interpreting Neural Netw. NLP*, Brussels, Belgium, Jan. 2018, pp. 56–65.

[23] N. D. Kanishka and S. P. Bagavathi, "Learning of generic vision features using deep CNN," in *Proc. 5th Int. Conf. Adv. Comput. Commun. (ICACC)*, Sep. 2015, pp. 54–57.

[24] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and J. D. Inman, "1D convolutional neural networks and applications: A survey," *Mech. Syst. Signal Process.*, vol. 151, Apr. 2019, Art. no. 107398.

[25] R. Yadav, "Light-weighted CNN for text classification," *CoRR*, vol. abs/2004.07922, 2020. [Online]. Available: https://arxiv.org/abs/2004.07922

[26] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning* (Springer Series in Statistics). New York, NY, USA: Springer, 2001.

SANJANASRI JP received the degree in August 2021, and the Ph.D. degree under guidance of Dr. Soman KP. She is currently serves as an Assistant Professor in computational engineering and networking (CEN) with the Multi-Disciplinary Research Center, Amrita School od Computing, Amrita Vishwa Vidyapeetham. She is strongly Associated with CEN, since 2010. She has published several conference and Journal papers so far. She has worked and did projects on several areas including, image processing, speech processing, the IoT devices such as Raspberry Pi, Arduino, and Jetson Nano. Her specific areas of research interests include machine learning, geometric deep learning, natural language processing, and big data.

VIJAY KRISHNA MENON is currently a Data Scientist (unicorn) and computational expert specialized in machine learning, deep learning, and other data driven techniques in big data. He works predominantly with big data on financial analytics, smart grid, and other data driven areas including NLP and text mining. He was a consultant to various companies providing his experience in NLP and data analytics. He is uniquely skilled in transforming complex mathematical models and algorithms to efficient computational implementations. He has a good research record on real-time analytics and streaming data of large volume and velocity and designing efficient data pipelines too. The recent areas he has worked includes but are not limited to text mining, financial forecasting, and stock market dynamics, data driven cybersecurity, computational epidemiology (modeling Covid-19), and spatiotemporal analytics in various domains.

SOMAN KP currently serves as the Head and a Professor with CEN and the Associate Dean of the Amrita School of Computing, Amrita Vishwa Vidyapeetham. He has more than 25 years of research and teaching experience in artificial intelligence and data science related subjects at the Amrita School of Engineering, Coimbatore. He has around 450 publications to his credit in reputed journals such as IEEE Transactions, IEEE Access, *Applied Energy*, and conference proceedings. He published four books namely *Insight into Wavelets, Insight into Data mining, Support Vector Machines and Other Kernel Methods, and Signal and Image Processing-The Sparse Way*. His book, insight into data mining was translated into Chinese. He is the most cited author in Amrita Vishwa Vidyapeetham in the area of artificial intelligence and data science (more than 5500 citations). He was listed among the Top-10 computer science faculty by DST, Government of India, from 2009 to 2013, the Career 360 and MHRD from 2017 to 2018, and also in the list of the most prolific authors in the world, prepared by Springer Nature. Under his guidance, CEN is running the M.Tech. degree in computational engineering and networking (data science) and the B.Tech. degree in computer science and engineering (artificial intelligence). He guided 12 Ph.D. students so far and currently guiding ten research scholars.

ATUL K. R. OJHA is currently pursuing the Ph.D. degree with the National University of Ireland, Galway. He is currently serves as an Adjunct Professor with the National University of Ireland. He is also the Co-Founder of Panlingua Language Processing LLP, India. He has published several papers in the area of NLP and conducted shared tasks and workshop in NLP in notable NLP conferences such as COLING, EMNLP, and so on. His area of research interests include data anlytics and NLP machine translation and localization, specch to speech and sign MT, digital humanities, metrics for MT evaluation and translation quality, corpus mining, semantic/syntactic parsing, machine learning, cognitive translation analysis, hate and aggressive speech, big data and computational linguistics, and artificial intelligence.