Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 218 (2023) 2604–2611

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Machine Learning and Data Engineering

BERT-CNN: Improving BERT for Requirements Classification using CNN

Kamaljit Kaur[a], Parminder Kaur[b]

[a]*Department of Computer Science, Guru Nanak Dev University, Amritsar*
[b]*Department of Computer Science, Guru Nanak Dev University, Amritsar*

Abstract

Requirements classification is considered a crucial task in requirements engineering. The analysis of functional and Non-functional requirements (NFRs) requires domain knowledge. NFRs are considered quality attributes, that hold critical information about the constraints upon which the success of software depends. Usually, requirements are expressed in natural language, and extraction and classification of such requirements become a challenging task. Various automatic techniques for requirements classification are exploited by existing studies. Recently, Transfer learning as a new deep learning model attracts the attention of researchers, which excelled in different Natural Language Processing tasks. Transfer learning-based BERT pre-trained model achieved more promising results than state-of-the-art approaches. This proposed research presents a Bidirectional Encoder-Decoder Transformer-Convolutional Neural Network (BERT-CNN) model for requirements classification. Then, the convolutional layer is stacked over the BERT layer for performance enhancement. This research work conducted an experiment on the PROMISE dataset of 625 requirements. The experiment results demonstrate that the proposed model performs better than the state-of-the-art baseline approach.

Keywords: Requirements classification, Machine learning, deep learning, Transfer learning, Convolutional neural network

1. Introduction

Software systems are engineered on the basis of software requirements specification (SRS) that consists of functional and non-functional requirements [1]. Functional requirements (FRs) are the sentences and propositions that specify the behaviour of the system and software components, also describing possible insights and events of the environment and corresponding required outputs. On the other hand, NFRs are the quality attributes that describes the behaviour of the software system and also provide the means to perform functional task [2]. Software quality NFRs includes performance, reliability, maintainability, portability, robustness, and security. The classification of requirements is important to identify specific behaviour of the system [3], [21].
Automatic identification of these requirements needs automatic processing of natural language requirements [21]. Recent studies [4],[5],[6],[7] have exploited the supervised and unsupervised techniques for automatic categorization of requirements. These studies reported promising results on heterogeneous datasets. Despite from the benefits, there are major limitations of approaches discussed by prior research: (i) the dimensionalities of these

10.1016/j.procs.2023.01.234

models are high, which decreases the performance of the model. (ii) Manual classification process is very time-consuming and labor-intensive.

In order to address aforementioned drawbacks of machine learning and traditional classification methods, deep neural network techniques are proposed and used by several researcher works, shows the promising results. Most machine learning techniques have exploited traditional word representation techniques such as Bag-of-word (BOW), Term-Frequency- Inverse Document Frequency (TF-IDF), noun phrases, and part-of-speech tag[4]. These traditional techniques suffer from sentence sequence ignorance. To address aforementioned issue, word embedding techniques are introduced such as word2vec and GloVe etc. These word embeddings have utilized neural network model that convert word into vectors. These vectors are fed to deep neural network as input, and improves the classification accuracy[8]**.** Recently, transfer learning based algorithms have been leveraged to tackle natural language processing related challenges [9],[22]. Transfer learning approaches trained on text corpus and fine-tuned them by adding more layers for classification tasks. Various language models[9], [10] are used in text classification task. This research work proposes BERT with CNN based deep learning model. BERT is a language model to extract features from requirements statements and CNN extract depth features at different levels. This proposed work use BERT-CNN for automatic classification requirements in the benchmark PROMISE NFRs dataset used in existing studies, and part of the RE'17 data challenge.
Our contribution is three fold:

- Building BERT-CNN model, which combines transfer learning and CNN to represent the depth feature of requirements statements.
- Evaluating our model on PROMISE dataset.
- Comparative analysis of proposed approach with existing approaches.

The rest of this paper is organized as follows: Section 2 discusses the prior related studies on requirements classification. Section 3 explains the proposed approach. Dataset description and experiment results are in Section 4 and 5 respectively. Finally, section 6 addresses conclusions and possible future works.

2. Related work

From past few years, the classification of functional and non-functional requirements in requirements engineering have been carried out by researchers[11].
Cleland-Huang et al. [12] have utilize information retrieval (IR) method to classify requirements into FRs and NFRs. They have identified indicator terms and classify NFRs by computing a probability value of its type. The authors have evaluated their approach on same dataset, which is also used by our research work. The reported high recall (81%) and low precision value (12%), indicating large number of false positive. Hussain et al. [3] have developed decision tree equipped with linguistic knowledge, to automatically detect NFRs. They have evaluated their approach on PROMISE NFRs dataset, and achieve 98% accuracy. Casamayor et al. [5] have proposed semi-supervised text categorization approach for automatic identification and classification of non-functional requirements. The authors have enhanced the performance of classifier by integrating Naive Bayes with expectation maximization. Empirical evidence on proposed approach shows that it achieves 70% of accuracy, which is better than supervised techniques. They concluded that semi-supervised techniques require less human efforts for labeling, than fully supervised techniques. Slankas and Williams [13] have developed NFRs locator to extract NFRs from several unstructured documents that are written in natural language such as SRS, user manual, and data agreements. The authors have also analyzed, extract and classify PROMISE NFRs into 14 categories. They have used sequential minimum optimizer (SMO), K-Nearest Neighbour (KNN) and Naïve Bayes (NB), and reported that SMO performs better than both algorithms.
Kurtanovic and Maalej [4] have developed Support Vector Machine approach for automatic classification of requirements into FRs and NFRs, and sub-classes of non-functional requirements. Their semi-supervised approach based on feature selection on lexical, metadata, and syntactic features to predict classes of NFRs of imbalance PROMISE dataset. The authors have also integrated user comment of Amazon product in order to maintain the balance between requirements classes. The authors have considered four major classes such as Usability, Security, Operational and Performance. They have used binary classifiers and reported 0.72 of precision and 0.90 of recall. Abad et al. [14] has proposed supervised and unsupervised machine learning techniques to improve the performance

of decision tree by [3] The authors have designed NFRs pre-processing approach with four machine learning algorithms (LDA, Biterm, Hierarchical k-mean, and hybrid and binarized naïve bayes). They have conducted experiments on 625 natural language software requirements. In order to remove inconsistencies and ambiguities, pre-processing techniques are used. The authors have compared the performance of classifier with or without pre-processing, and achieve F1-score of 45% and 90% respectively. Winkler and Vogelsang [15] have introduced an approach to classify software requirements as either "requirements" or "information". They have developed CNN based tool for aforementioned classification problem, and achieves 82% of F1-score. Navarro-Almanza et al. [7] have also utilized CNN based deep learning approach to classify software requirements of PROMISE dataset. They concluded that their approach achieved 80% of precision, 78% of recall, and 77% of F1-score. Dekhtyar and Fong [16] have proposed deep learning based model for requirements classification and reported 92% of F1-score. Tiun et al. [17] have employed word embedding with deep neural network to classify software requirements. The reported results have illustrated that FastText word representation technique performs better than word2Vec. Amasaki and Leelaprute [18] performed set of experiments to compare different vectorization methods for classification of NFRs. They reported that Doc2Vec vector performs better than traditional ones (e.g. BOW and TF-IDF). Baker et al. [19] have compared two neural networks such as Artificial Neural Network (ANN) and CNN. The authors have evaluated their approach on natural language requirements and concluded that CNN significantly performs better than ANN. Gnanasekaran et al. [20] have applied Recurrent Neural Network (RNN) and compare with [19], and demonstrate that RNN performs better than CNN. Hey et al. [21] have introduced transfer learning approach for requirements classification and achieves 94% of F1-score on PROMISE dataset. Kaur and Kaur [22] have proposed self-attention based Long- Short term memory (LSTM) model for requirements classification on natural language software requirements. The authors have concluded that their proposed approach performs better with 96% of F1-score.
Furthermore, the applicability and performance of requirements classification can be improved by adding more layers to transfer learning model. By following the same vein, this paper proposes the hybrid model called BERT-CNN which combines BERT and CNN to achieve better classification results. For this CNN layers are added after the BERT layer, to capture important features from requirements statements.

3. BERT-CNN method

In this section, the research work introduces the overall process of BERT-CNN for requirements classification. In recent years, transfer learning reported remarkable results in natural language processing task such as sequence classification, entity recognition and question answering etc [21]. Various language models such as ELMo, word2Vec, and GloVe are used to extract features for downstream task and classify them[23]. These traditional models suffer from polysemic words, and unable to maintain contextual relationship between the words [23]. To tackle this problem, transformer based architecture is introduced by [24]. These transformer based architecture used encoder-decoder, self-attention mechanism and feed-forward network. Where, attention mechanism focuses on the important features of the sentence. Then, Bidirectional Encoder representation from Transformer has been introduced by (Devlin et al., 2019) [9]. The authors have introduced two BERT model i.e. Base and Large. BERT (base): Encoder layer= 12, Attention heads=12, Hidden units=768, Total parameters=110M. Whereas, BERT (Large): Encoder layer= 24, Attention heads=16, Hidden units= 1025, Total parameters=340M. In contrast to previous pre-training models, BERT is able to handle bidirectional context, in which two methods are used such as Masked Language Model (MLM) and Next Sentence Prediction (NSP).
Convolutional neural network is a deep learning model that can be largely used in text classification because it extracts low level features and controlling the length of dependency [25]. In order to extract abstract representation of text, multiple Convolutional layers can be used. Then, pooling layer is used extract significant features. Navarro-Almanza et al. [7] have used traditional representation technique i.e word2Vec. However, the main limitation of word2Vec is that, it does not retain the sequence between the texts, while generating the word vectors [23]. As a result, this proposed work explored the importance of pre-trained model BERT as a word representation for requirements statements.
The main focus of this research work is to find the contextual information from sequence of the related words. Unlike word2vec and GloVe, BERT retains the sequence while taking the context into account. Transfer learning also boosts the classifier accuracy. Section 3.1, explains the details of BERT-CNN model which contain two distinctive subsections.

3.1 BERT-CNN Model

The learning process flow of BERT-CNN is presented in figure 1. The proposed framework consists of four layers: embedding layer, Convolutional layer, pooling layer and output layer. BERT model is used to generate word vectors of input sequence, and also create a primary input matrix. Then, on the basis of input vector matrix, feature maps are generated by Convolutional layer. The output of convolutional layer is given to pooling layer and pooling layer extract the max value feature vectors. Ultimately, the feature vectors are fed to fully connected layer for classification. The detail description of each layer is given as follows:

3.1.1 Embedding Layer

To generate the input sentence matrix, BERT as embedding layer creates input for convolutional layer, where each sentence is converted into word vector. In embedding layer, BERT utilized to store contextual representation of input requirement sentence. This research work used pre-trained masked language modelling to obtain context dependent word representation. BERT can perform two tasks such as generate input sequence as word vectors and fine-tuned for classification task. From the literature, it can be concluded that BERT performs better in requirements classification task. Therefore, it is assumed that using BERT as word representation technique can enhance the classification accuracy of the model. In summary, the sentence length is represented by s; d represents the matrix dimensions and then sequence matrix is denoted as $S \in R^{s\times d}$

3.1.2 Convolutional layer

In BERT-CNN, Convolutional Neural Network is used to extract the local features, which results reduction in the dimension of the data. The Convolutional layer is primary building block of CNN. This layer computes Convolutional function on semantic vectors, by convolving the data with N randomly generated filters. The proposed research work set uniform length of filters (h) and dimensionality of word vector (d). Moreover, dot product between two sequence metrics is performed. Assuming s is sequence matrix $S \in Q^{s\times d}$, Convolutional filter $H \in Q^{h\times d}$ is performed to generate its submatrix indicated as $S_{i:j}$. Then, the Convolutional operation is performed repeatedly on sequence matrix to obtained output sequence matrix $C \in Q^{s-h+1\times d}$.

$$C_i = a\,[w_c^T S_{i:i+m-1} + b) \in Q^i \qquad (1)$$

Here, i=s-h+1, and b is the bias vector and a is the non-linear activation function of Convolutional operation. Rectified Linear Unit (ReLU) is used as non-linear activation function because it can improve the learning dynamics of the network and significantly reduce the number of iterations required for the convergence in deep neural networks.

3.1.3 Pooling layer

The prominent goal of pooling layer is to summarize the results obtained from Convolutional layer and extract the more representative section of feature maps. Different feature maps are produced by different filter sizes, and pooling function is used to produce fixed-sized vector. This paper also utilizes the pooling function to extract the the important feature from feature maps. For each filter, the output feature is the representative value of the feature set, and max pooling is used over for all output (feature maps) created by convolutional layer. Then, all features generated by max pooling layer are concatenated to obtained final vector representation of requirement statement.

3.1.4 Fully connected layer and Loss function

The concatenated features i.e. output of pooling layer can be utilized as input to fully connected layer with the q neurons as the output layer of the model. Then, softmax function is used as the output function of the model, and its formula is:

(2)

$$M_i = \frac{\exp(x)}{\sum_{j=1}^{c} \exp(x_j)}$$

Where x= $w_c^n S_{i:i+m-1.}$ Moreover, this research paper makes use of cross-entropy function as the loss function for the training model. The formula specifies the difference between the obtained distribution ($\widehat{M}_i(c)$) and the real classification distribution $M_i(c)$.

$$L_{loss} = \sum_{S \in T} \sum_{i=1}^{L} \widehat{M}_i(C) \log(M_i(C)) \quad (3)$$

Where n, L indicates training set and class respectively.

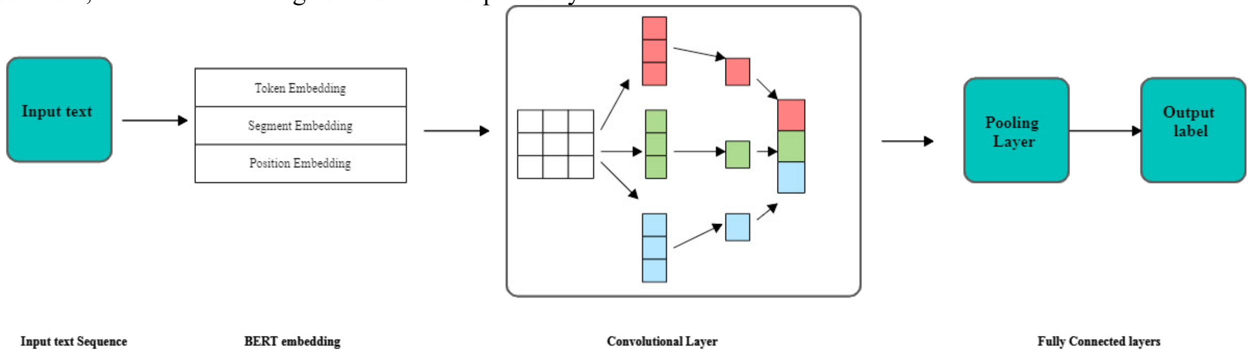


Fig 1: Architecture of BERT-CNN Model

4. Experiments

The main objective of this paper is to investigate the effect of BERT with CNN on the accuracy of requirements classification. To prove the effectiveness of aforementioned framework, this research makes use of PROMISE dataset and results are obtained by using evaluation metrics are also explained in detail.

4.1 Dataset

Standard and popular dataset is used in this paper to compare the comprehensive analysis of effectiveness of proposed model. The dataset consists of 625 requirements collected from 15 projects by students. The dataset include 255 FRs and 370 NFRs [21].Table 1 shows the class distribution in the dataset. Each requirement is labelled with only one class either FRs or one of the 11 subclasses of NFRs.

Table 1: Distribution of classes in dataset [21]

NFRs Label	Full Name	Number of requirements
A	Availability	21
L	Legal	13
LF	Look and Feel	38
MN/MT	Maintainability/Maintenance	17
O	Operability	62
PE	Performance	54
SC	Scalability	21
SE	Security	66
US	Usability	67
FT	Fault Tolerance	10
PO	Portability	1
F	Functional	255
Total		625

4.2 Evaluation measure

The proposed approach can be evaluated by using standard evaluation metrics. By taking inspiration from existing research works, three evaluation metrics precision, recall and F1-score are used.

Precision: Precision is ratio of model correctly classified the requirements to corresponding label to the total number of requirements of class. The formula for prediction measure is given below:

$$Precision = \frac{T_p}{T_p + F_p} \tag{4}$$

Recall: It is defined as the number of correctly predicted classes over the total relevant type of requirements classes.

$$Recall = \frac{T_p}{T_p + F_n} \tag{5}$$

F-Measure: F-measure is calculated as the harmonic mean of the precision and recall measure.

$$F - Measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{6}$$

Where T_p denoted as true positive, also indicated that number of correct predictions, F_p denoted as false positive, also indicates the non-relevant classification, F_n denoted as false negative, also that indicates relevant item which is not predicted by the model.

.

4.3 Parameter settings

The performance of the proposed approach can be evaluated under different settings as shown in table 2. Tensorflow is a python library that can implement deep learning algorithm. The implementation of BERT-CNN model is done by Tensorflow with BERT[*]. For the sake of comparison, we choose BERT cased same as implemented by [21]. The batch size and dropout for training are set to 32. The model is train with the help of back propagation algorithm with the Adam stochastic optimizer [26] along with learning and decay rate of 2e-5. The categorical cross-entropy loss is used as the loss function for multi-class classification. To prevent the overfitting, ten-fold stratified cross-validation technique is used in the training process.

5. Results

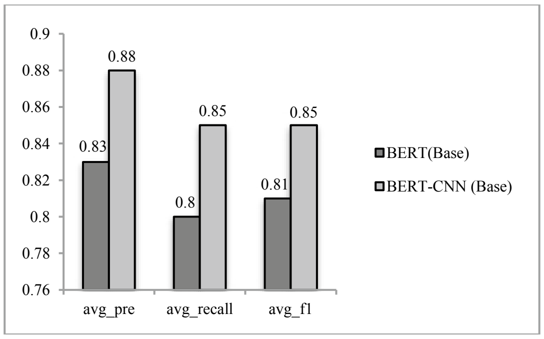
The proposed research work evaluates the performance of BERT-CNN when classifying requirements in subcategories of NFRs on PROMISE dataset. This paper employed ten-fold stratified cross-validation technique to measure the performance of proposed model. This research work perform comparative analysis of proposed approach with other baseline approaches such as [4], [14], [16], and [21]. Table 2 shows the proposed model outperforms the existing approaches and also perform better than BERT with weighted F1-score of 85% for frequent classes. This proves that CNN layer extract more semantic information and leads to performance enhancement.

This research work evaluates the performance of BERT-CNN for multi-class classification. The proposed model train multi-class classifier frequent NFRs subclasses (Operability, Performance, Security, and Usability). From the table 2, it can be concluded that BERT-CNN outperforms BERT for multi-class classification. Figure 2 shows the comparison of existing model and proposed model on the basis of average precision, recall and f-measure.

[*] https://tfhub.dev/google/collections/bert/1

Table 2: Comparison of results obtained on PROMISE dataset for multi-class classification, O: Operability, PE: Performance, SE: Security, US: Usability. #P: Precision, #R: Recall,#F1: F-measure, #A: average F1

Approach	**Parameter**	**O**			**PE**			**SE**			**US**			
		#P	**#R**	**#F1**	**#P**	**#R**	**#F1**	**#P**	**#R**	**#F1**	**#P**	**#R**	**#F1**	**#A**
K&M$_{bin}$	(500 best features)	0.72	0.73	0.71	0.87	0.81	0.82	0.74	0.81	0.74	0.80	0.71	0.74	0.75
NoRBERT$_{bin}$	(base, ep=10)	0.80	0.53	0.64	0.88	0.80	0.83	0.93	0.82	0.87	0.81	0.69	0.74	0.77
K&M$_{multi}$	(500 best features)	0.47	0.62	0.51	0.81	0.74	0.76	0.64	0.53	0.56	0.70	0.66	0.64	0.61
NoRBERT$_{multi}$	(base, ep=32) (10-fold)	0.79	0.73	0.76	0.88	0.78	0.82	0.89	0.85	0.87	0.78	0.84	0.81	0.82
BERT-CNN$_{multi}$	**(base, ep=10) (10-fold)**	**0.85**	**0.85**	**0.83**	**0.93**	**0.81**	**0.86**	0.89	**0.89**	0.87	**0.85**	**0.84**	**0.84**	**0.85**
NoRBERT$_{multi}$	(large, ep=32) (10-fold)	0.83	0.71	0.77	0.90	0.81	0.85	0.91	**0.91**	0.91	0.86	0.82	0.81	0.84
BERT-CNN$_{multi}$	(large, ep=10) (10-fold)	**0.86**	**0.85**	**0.85**	**0.96**	0.78	0.85	0.91	0.89	0.89	**0.86**	**0.93**	**0.89**	**0.87**



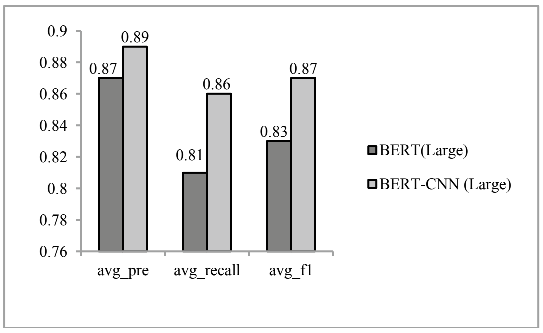


Fig 2: Comparison analysis of BERT vs. BERT-CNN on the basis of average precision, recall and f-measure

6. Conclusion

This research work presented an approach to requirements classification based on BERT-CNN in natural language processing. In order to get the important feature from requirement statements this paper employs CNN in task-specific layers of BERT. Experiments are conducted on PROMISE dataset to evaluate the performance of proposed approach.

The evaluation includes multi-class classification, where multi-class classification considers four frequent classes such as Operability, Performance, Security and Usability. On comparative analysis with some existing baseline methods, it can be concluded that BERT-CNN model performs better than simple BERT.
In addition, this paper combined BERT with CNN, in future work; we will explore the research field with integration of attention mechanism with proposed model. In addition, new methods are also applied to the field of requirements classification. Future work mainly includes (i) using other language models for the requirements classification. (ii) Use one or more attention mechanism in order to improve the classification accuracy. (iii) Using other deep learning models for requirements classification.

References

[1] Tóth, László, and László Vidács. (2019) "Comparative Study of The Performance of Various Classifiers in Labeling Non-Functional Requirements." *Information Technology and Control* 48, no. 3, pp:432-445.doi: 10.5755/j01.itc.48.3.21973.

[2] Haque, Md Ariful, Md Abdur Rahman, and Md Saeed Siddik. (2019) "Non-functional requirements classification with feature extraction and machine learning: An empirical study." In *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*. pp. 1-5. IEEE/ doi: 10.1109/ICASERT.2019.8934499.

[3] Hussain, Ishrar, Leila Kosseim, and Olga Ormandjieva. (2008) "Using linguistic knowledge to classify non-functional requirements in SRS documents." In *International Conference on Application of Natural Language to Information Systems*. pp. 287-298. Springer, Berlin, Heidelberg. doi: 10.1007/978-3-540-69858-6_28.

[4] Kurtanović, Zijad, and Walid Maalej. (2017) "Automatically classifying functional and non-functional requirements using supervised machine learning." In *2017 IEEE 25th International Requirements Engineering Conference (RE)*. pp. 490-495. IEEE. doi: 10.1109/RE.2017.82.

[5] Casamayor, Agustin, Daniela Godoy, and Marcelo Campo. (2010) "Identification of non-functional requirements in textual specifications: A semi-supervised learning approach." *Information and Software Technology* 52, no. 4: 436-445. doi: 10.1016/j.infsof.2009.10.010.

[6] Younas, Muhammad, Dayang NA Jawawi, Imran Ghani, and Muhammad Arif Shah. (2020) "Extraction of non-functional requirement using semantic similarity distance." *Neural Computing and Applications* 32, no. 11: 7383-7397. doi: 10.1007/s00521-019-04226-5.

[7] Navarro-Almanza, Raul, Reyes Juarez-Ramirez, and Guillermo Licea. (2017)"Towards supporting software engineering using deep learning: A case of software requirements classification." In *2017 5th International Conference in Software Engineering Research and Innovation (CONISOFT)*. pp. 116-120. IEEE. doi: 10.1109/CONISOFT.2017.00021.

[8] Wang, Yequan, Minlie Huang, Xiaoyan Zhu, and Li Zhao. (2016) "Attention-based LSTM for aspect-level sentiment classification." In *Proceedings of the 2016 conference on empirical methods in natural language processing*. pp. 606-615. doi: 10.18653/v1/d16-1058.

[9] Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. (2018) "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv: 1810.04805* .

[10] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le. (2019) "XLNet: Generalized autoregressive pretraining for language understanding." *Adv. Neural Inf. Process. Syst.*, vol. 32, no. NeurIPS, pp. 1–18.

[11] Dias Canedo, Edna, and Bruno Cordeiro Mendes. (2020) "Software requirements classification using machine learning algorithms." *Entropy* 22, no. 9: 1057. doi: 10.3390/e22091057.

[12] Cleland-Huang, Jane, Raffaella Settimi, Xuchang Zou, and Peter Solc. (2007) "Automated classification of non-functional requirements." *Requirements engineering* 12, no. 2: 103-120. doi: 10.1007/s00766-007-0045-1.

[13] Slankas, John, and Laurie Williams. (2013) "Automated extraction of non-functional requirements in available documentation." In *2013 1st International workshop on natural language analysis in software engineering (NaturaLiSE)*, pp. 9-16. IEEE. doi: 10.1109/NAturaLiSE.2013.6611715.

[14] Abad, Zahra Shakeri Hossein, Oliver Karras, Parisa Ghazi, Martin Glinz, Guenther Ruhe, and Kurt Schneider. (2017) "What works better? a study of classifying requirements." In *2017 IEEE 25th International Requirements Engineering Conference (RE)*. pp. 496-501. IEEE. doi: 10.1109/RE.2017.36.

[15] Winkler, Jonas, and Andreas Vogelsang. (2016) "Automatic classification of requirements based on Convolutional neural networks." In *2016 IEEE 24th International Requirements Engineering Conference Workshops (REW)*. pp. 39-45. IEEE. doi: 10.1109/REW.2016.16.

[16] Dekhtyar, Alex, and Vivian Fong. (2017) "Re data challenge: Requirements identification with word2vec and tensorflow." In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pp. 484-489. IEEE. doi: 10.1109/RE.2017.26.

[17] Tiun, S., U. A. Mokhtar, S. H. Bakar, and S. Saad. (2020). "Classification of functional and non-functional requirement in software requirement using Word2vec and fast Text." In *journal of Physics: conference series*, vol. 1529, no. 4. p. 042077. IOP Publishing. doi: 10.1088/1742-6596/1529/4/042077.

[18] Amasaki, Sousuke, and Pattara Leelaprute. (2018) "The effects of vectorization methods on non-functional requirements classification." In *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pp. 175-182. IEEE. doi: 10.1109/SEAA.2018.00036.

[19] Baker, Cody, Lin Deng, Suranjan Chakraborty, and Josh Dehlinger. (2019) "Automatic multi-class non-functional software requirements classification using neural networks." In *2019 IEEE 43rd annual computer software and applications conference (COMPSAC)*, vol. 2, pp. 610-615. IEEE.doi: 10.1109/COMPSAC.2019.10275.

[20] Gnanasekaran, Rajesh Kumar, Suranjan Chakraborty, Josh Dehlinger, and Lin Deng. (2021) "Using Recurrent Neural Networks for Classification of Natural Language-based Non-functional Requirements." In *REFSQ Workshops*.

[21] Hey, Tobias, Jan Keim, Anne Koziolek, and Walter F. Tichy. (2020) "NoRBERT: Transfer learning for requirements classification." In *2020 IEEE 28th International Requirements Engineering Conference (RE)*, pp. 169-179. IEEE,. doi: 10.1109/RE48521.2020.00028.

[22] Kaur, Kamaljit, and Parminder Kaur. (2022) "SABDM: A self-attention based bidirectional-RNN deep model for requirements classification." *Journal of Software: Evolution and Process* : e2430. doi: 10.1002/smr.2430.

[23] Rodríguez Medina, Samuel. (2019) "Multi-Label Text Classification with Transfer Learning for Policy Documents: The Case of the Sustainable Development Goals."

[24] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. (2017) "Attention is all you need." *Advances in neural information processing systems* 30.

[25] Liu, Gang, and Jiabao Guo. (2019) "Bidirectional LSTM with attention mechanism and convolutional layer for text classification." *Neurocomputing* 337: 325-338. doi: 10.1016/j.neucom.2019.01.078.

[26] Kingma, Diederik P., and Jimmy Ba. (2014) "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980*