**DOI: 10.15514/ISPRAS-2024-36(1)-8**



# Deep Learning for Non-functional Requirements: A Convolutional Neural Network Approach

[1] *S. E. Martinez García,* ORCID: 0009-0006-3907-8384 <*estefmartinezgarcia@gmail.com*>
[2] *C. A. Fernández-y-Fernández,* ORCID: 0000-0002-1586-8772 <*caff@mixteco.utm.mx*>
[2] *E. G. Ramos Pérez,* ORCID: 0000-0001-8337-4195 <*erik@mixteco.utm.mx*>

[1] *División de Estudios de Posgrado, Universidad Tecnológica de la Mixteca, Huajuapan de León, Oax., México.*
[2] *Instituto de Computación, Universidad Tecnológica de la Mixteca, Huajuapan de León, Oax. México.*

**Abstract.** The Requirements Engineering (ER) phase plays a critical role in software development, as any shortcomings during this stage can lead to project failure. Analysts rely on Requirements Specification (RS) to define a comprehensive list of quality requirements. The process of requirements classification, within RS, involves assigning each requirement to its respective class, presenting analysts with the challenge of accurate categorization. This research focuses on enhancing the classification of non-functional requirements (NFR) using a Convolutional Neural Network (CNN). The study also emphasizes the significance of preprocessing techniques, the implementation of sampling strategies, and the incorporation of pre-trained word embeddings such as Fasttext, Glove, and Word2vec. Evaluation of the proposed approach is performed using metrics like Recall, Precision, and F1, resulting in an average performance improvement of up to 30% compared to related work. Additionally, the model is assessed concerning its utilization of pre-trained word embeddings through ANOVA analysis, providing valuable insights into its effectiveness. This study aims to demonstrate the utility of CNNs and pre-trained word embeddings in the classification of NFRs, offering valuable contributions to the field of Requirements Engineering and enhancing the overall software development process.

**Keywords:** deep learning; non-functional requirements; convolutional neural network; requirements engineering.

**For citation:** Martinez-García S. E., Fernández-y-Fernández C. A., Ramos-Pérez E. G. Deep Learning for Non-functional Requirements: A Convolutional Neural Network Approach. Trudy ISP RAN/Proc. ISP RAS, vol. 36, issue 1, 2024. pp. 131-142. DOI: 10.15514/ISPRAS-2024-36(1)-8.

**Full text:** Martinez García S. E., Fernández-y-Fernández C. A., Ramos Pérez E. G. Classification of Non-Functional Requirements Using Convolutional Neural Networks. Programming and Computer Software, 2023, Vol. 49, No. 8, pp. 705–711. DOI: 10.1134/S0361768823080133.

# Глубокое обучение при выработке нефункциональных требований: подход на основе сверточных нейронных сетей

[1] *С. Э. Мартинес Гарсия*, ORCID: 0009-0006-3907-8384 <*estefmartinezgarcia@gmail.com*>
[2] *К. А. Фернандес-и-Фернандес*, ORCID: 0000-0002-1586-8772 <*caff@mixteco.utm.mx*>
[2] *Э. Г. Рамос Перес*, ORCID: 0000-0001-8337-4195 <*erik@mixteco.utm.mx*>

[1] Аспирантура Технологического университета Миштека,
*Уахуапан де Леон, Оахака, Мексика.*
[2] *Институт вычислений Технологического университета Миштека,*
*Уахуапан де Леон, Оахака, Мексика.*

**Аннотация.** Фаза разработки требований (ER) играет решающую роль в разработке программного обеспечения, поскольку любые недостатки на этом этапе могут привести к провалу проекта. Аналитики полагаются на спецификацию требований (RS) для определения полного списка требований к качеству. Процесс классификации требований в рамках RS включает отнесение каждого требования к соответствующему классу, что ставит перед аналитиками задачу точной классификации. Данное исследование направлено на улучшение качества классификации нефункциональных требований (NFR) на основе применения сверточной нейронной сети (CNN). В исследовании также подчеркивается важность методов предварительной обработки, реализации стратегий выборки и включения предварительно обученных векторных представлений слов, таких как Fasttext, Glove и Word2vec. Оценка предлагаемого подхода выполняется с использованием таких метрик, как Recall, Precision и F1, что приводит к среднему улучшению производительности до 30% по сравнению с другими подходами. Кроме того, модель оценивается в отношении использования предварительно обученных векторных представлений слов с помощью анализа ANOVA, предоставляя ценную информацию о ее эффективности. Это исследование направлено на то, чтобы продемонстрировать полезность CNN и предварительно обученных векторных представлений слов в классификации NFR, предлагая ценный вклад в области инженерии требований и улучшая общий процесс разработки программного обеспечения.

**Ключевые слова:** глубокое обучение; нефункциональные требования; сверточная нейронная сеть; инженерия требований.

**Для цитирования:** Мартинес-Гарсия С. Э., Фернандес-и-Фернандес К. А., Рамос-Перес Э. Г. Глубокое обучение при выработке нефункциональных требований: подход на основе сверточных нейронных сетей. Труды ИСП РАН, том 36, вып. 1, 2024 г., стр. 131–142 (на английском языке). DOI: 10.15514/ISPRAS–2024–36(1)–8.

**Полный текст:** Мартинес Гарсия С. Э., Фернандес-и-Фернандес К. А., Рамос Перес Э. Г. Классификация нефункциональных требований на основе сверточных нейронных сетей. Programming and Computer Software, 2023, т. 49, № 8, стр. 705–711 (на английском языке). DOI: 10.1134/S0361768823080133.

## *1. Introduction*

During the initial phases of the software development life cycle, regardless of the model that is intended to be followed, the requirements phase is declared as a key piece to achieving a successful development [1-3, 8, 31]. If the requirements are not discovered and defined correctly in this early phase, failures arise during development, which promotes that the final delivery is that of incomplete software, that is to say, that it does not do what it should do, adding to that the Established times are not met and are extended, which will cause previously estimated costs to rise [2, 24].

For this reason, this phase is considered vital since the correct execution of the activities will prevent failure of software development [2, 5, 27, 30]. To combat this problem, analysts have used Requirements Engineering (RE) [4, 17, 23], which is characterized by producing a list of quality requirements as a final result. When carrying out the classification of requirements, there are difficulties of interpretation and identification [13, 18] (an inherent characteristic of natural language

[9, 25, 28]) to determine to which class each of these belong, since there are functional (FR) and non-functional (NFR) requirements, of which the latter contain subclasses. In addition to this difficulty, the extensive list of requirements is also presented, which can number in the thousands, so it would be a job that takes too much time and effort [12, 20, 26].

In this research machine learning techniques are used to apply them in the RE in the Requirements Specification (RS) activity, specifically on the classification of requirements. Said activity consists of identifying the class of requirement to which it belongs or simply differentiating between a requirement and information [13-14]. In particular, it will focus on NFRs, since they are frequently discriminated against because they are considered of little or no importance for software development, as well as the lack of knowledge to identify them [6-7, 11, 16].

## *2. Background*

### 2.1 Data set

The NFR quality attributes data set, also known as the PROMISE [29] corpus, is a compilation of requirements specifications for 15 software projects developed by students at DePaul University as a term project for a course in Requirements Engineering; The language of the content is in English. The data set consists of 326 NFRs and 358 FR requirements. The NFR dataset lends itself, for purposes of this research, to the multi-label classification of various types of NFR requirements.

### 2.2 Input format for classifier

A classifier expects the data to be in the form of a list of strings of requirements (referred to as examples) in the form of a vector of one-hot words *one-hot* and an attached list of vectors representing the associated requirement class.

### 2.3 Sampling Strategies

The objective of the sampling strategies is to avoid the imbalance in the distribution of the class that the datasets constantly present, this imbalance causes the automatic learning algorithms to have low performance in the minority class; since the cost of misclassifying it is usually much higher than the cost of other misclassifications [10, 19, 32]. Therefore, when selecting the Promise data set, it was observed that the distribution of the set is unbalanced, so it is appropriate to use this strategy.

### 2.4 Evaluation metrics for classifier performance

To evaluate the model, the same metrics used by [10, 15, 33] will be taken as a reference, since in addition to establishing the improvement of the work done, they measure the performance of the model with respect to the correct predictions it makes. They are briefly shown below:

- **Accuracy**. It is the total percentage of cases classified correctly.
- **Recall**. It is the number of data correctly identified as positive out of the total number of true positives.
- **Score F1**. It can be interpreted as a weighted average of precision and *recall*, where an F1 score reaches its best value at 1 and its worst score at 0.
- **Precision**. Accuracy is the ratio of correct predictions to the total number of predicted correct predictions.

## 2.5 Related work

The aim of this section is to explore, identify and improve the CNN preprocessing and configuration bases proposed by [10, 15, 33]. The key point for the classification of NFR is that its nature is multiclass.

- In [33] does not use the *Promise* dataset, and performs binary classification. In their research, they do not show which configuration was used, and they do not mention any type of validation used, in addition, the authors mention that because the data set they used contains little information, the requirements classification obtained an accuracy of 73.
- [15], perform multiclass classification, if you have the *Promise* data set, but use all 12 classes, that is, both Functional and Non-functional Requirements. He used the embedded fastText model. He got 80 %, using cross- validation with parameters of $k = 10$ and applying the optimizer *AdamOptimizer*.
- [10], performs multiclass classification, also occupies *Promise* and implements experimentation on NFR, especially on 9 classes, since the other two that belong to this type of requirements have few examples, which that prevents the use of sampling strategies; therefore, at least two examples are required to carry it out. It does not refer to what type of strategy obtained the results it presents, so for the purposes of this paper both random oversampling (ROS) and random undersampling (RUS) were tested, however, in this section only the optimal result, which was ROS, is presented, to see the results of the experimentation with RUS see the section in the index. He used the embedded fastText and Word2Vec models. The accuracy result was 80.4% with the *Word2Vec* vectorization method, this was the highest compared to the pre-trained Fasttext matrix and a random weights matrix.

## *3. Experimentation and results*

The experimentation seeks to determine the influence of the text preprocessing, the vectorization of the data, the ROS sampling strategy, the implementation of the pre-trained embedded matrices of Word2Vec, fastText, and Glove in the embedded layer of the CNN, and the hyper parameterization of its subsequent layers. The key point for the classification of NFR is that its nature is multiclass, so it was determined, based on previous experiments, to build models combining the improvements made in each previous experiment and, if possible, adapt them to this model. rating with CNN to try to increase rating metrics. So, the points to consider are the following:

- **Preprocessing**. Whether or not to include pre-cleaning of the Promise dataset, as well as applying lemmatization to words. It is also tested with 3 types of vectorizers: *TF-IDF*, *Tokenizer*, and *CountVectorizer*.
- **Hyperparameterization in the data partition**. Implementation of the sampling strategy ROS proposal.
- **CNN architecture**. Starting with the base architecture including or not the weights of the pre-trained matrices.

## 3.1 Results and comparison

The results shown below are presented gradually; that is, it indicates the way in which by adding the proposed techniques to a base CNN architecture, the classification performance improves.

Now we show the concentration of results obtained from experimentation with a progressive integration of proposals to improve the classification of NFR using NFR. First, in Table 1, where an average of *Recall* of 0.11, *Precision* of 0.01 and *F1* of 0.02 was achieved for the vectorizer test. TF-IDF; On the other hand, with the Tokenizer vectorizer, an average of 0.44 was obtained for *Recall*, 0.46 for *Precision* and 0.44 for the metric *F1*. On the other hand, with the CountVectorizer

vectorizer, it can be seen that the training of the network with that tool gained a *Recall* with 0.59, *Precision* with 0.53, and *F1* with 0.54. The second part of the experimentation is integrating the previous cleaning and lemmatization, in addition to making the hyper parameterization in the data partition; therefore, the averages resulting from this stage are those shown in Table 2. An increase in *accuracy* is observed, however, when directing attention to the average of the metrics, it is examined that when applying these proposals, a notable improvement is found for the case of the vectorizer *CountVectorizer*, where emphRecall has a 0.70, *Precision* with a 0.69 and *F1* with a 0.67. Therefore, the classifier has shown a particular behavior on *CountVectorizer*, both in this test and in the previous one. Finally, one more proposal is added, which is to train the model with the weights of the pre- trained matrix *Word2vec*, since it was a common denominator among the proposals of [10, 15, 33]. The averages obtained can be seen in Table 3, *Recall* with 0.72, *Precision* with 0.74, and *F1* with 0.72; Therefore, an increase in the averages could be observed using said weight matrix. Therefore, when observing the effect of each of these proposals implemented on the classifier, it was possible to determine that for the following experiments, it is convenient to use CountVectorizer since with the 2 vectorizers it can be seen that they do not promote an improvement with respect to the processing of the data. In addition to continuing to use the proposals based on what has been observed in the experiments already carried out.

*Table 1. Base model results*

| Test | Vectorizer | Average | | |
|---|---|---|---|---|
| | | Recall | Precision | F1 |
| 1 | TF-IDF | 0.11 | 0.01 | 0.02 |
| 2 | Tokenizer | 0.44 | 0.46 | 0.44 |
| 3 | CountVectorizer | 0.59 | 0.53 | 0.54 |

*Table 2. Base model results with integration of two proposals*

| Test | Vectorizer | Average | | |
|---|---|---|---|---|
| | | Recall | Precision | F1 |
| **1** | TF-IDF | 0.11 | 0.02 | 0.04 |
| **2** | Tokenizer | 0.41 | 0.45 | 0.39 |
| **3** | CountVectorizer | 0.70 | 0.69 | 0.67 |

*Table 3. Base model results with integration of three proposals*

| Test | Vectorizer | Average | | |
|---|---|---|---|---|
| | | Recall | Precision | F1 |
| 1 | TF-IDF | 0.11 | 0.02 | 0.04 |
| 2 | Tokenizer | 0.42 | 0.41 | 0.40 |
| 3 | CountVectorizer | 0.72 | 0.74 | 0.72 |

*CNN with the implementation of pre-trained matrices and ROS sampling strategy:* For this experiment, a *Dropout* layer was added, after the embedded layer, as well as another set of a convolution followed by a *MaxPooling* layer.

Table 4 shows the results of the metrics obtained by implementing each of the different pre-trained embedded matrices. The text preprocessing, the ROS sampling strategy, and the new architecture were used for this training that was carried out with 100 epochs. However, the data vectorization was the important factor in improving the results. For this, the *CountVectorizer* library was used, which in addition to obtaining properties such as eliminating *stopwords*, calculating the frequency of words among others, also has the *ngram range* argument. This argument determines the lower and upper bound of the range of n values for different words, called *n-grams* [21-22]. Hence, this argument was essential and was considered to obtain a vectorization of words that helps to make

sense of each of the requirements according to their context. The *ngram range* that allowed the results shown in Table 4 to be obtained was (1, 4) for fastText and Word2Vec and (1, 2) for Glove. It is important to highlight that tests were carried out without *ngram range* and with ranges of (1,1), (1,2), (1,3), and (1,4), for each experimentation with the embedded matrices selected. Therefore, 5 tests were carried out for each script.

Making the comparison in Table 5, it can be seen how the base proposal with 100 epochs reflected a notable increase with respect to the optimal results obtained by [10] with 140 epochs. To determine if there really is an improvement with respect to the initial configurations, it has been proposed to carry out a statistical analysis in the following section.

*Table 4. CNN results with 100 epochs for NFR classification using 3 types of embedded arrays*

| Embedded matrix | Average | | |
|---|---|---|---|
| | Recall | Precision | F1 |
| Word2vec | 0.88 | 0.90 | 0.88 |
| FastText | 0.83 | 0.85 | 0.83 |
| Glove | 0.82 | 0.79 | 0.79 |

## 3.2 Evaluation of the proposals implemented to the CNN model with cross-validation k-fold by analysis of variance

To evaluate the model implemented for the CNN with the different 3 pre-trained matrices used, the analysis of variance (ANOVA) was performed. Taking the averages of *F1* resulting from cross-validation training *k-fold* with *k*=10, since said metric represents the average between *Recall* and *Precision*. The goodness of *F1* is useful when there is an unequal distribution in the classes, this being the case of the data set being used, for this reason, it was decided to perform ANOVA on that metric.

*Table 5. Results of the metrics obtained from the optimal preprocessing and architecture of [10] vs. results of the preprocessing and base architecture proposed in this paper*

*(a) Results obtained by [10] for 140 epochs*

| Embedded matrix | Average | | |
|---|---|---|---|
| | Recall | Precision | F1 |
| Random | 0.66 | 0.66 | 0.66 |
| Word2vec | 0.75 | 0.79 | 0.77 |
| FastText | 0.73 | 0.76 | 0.76 |

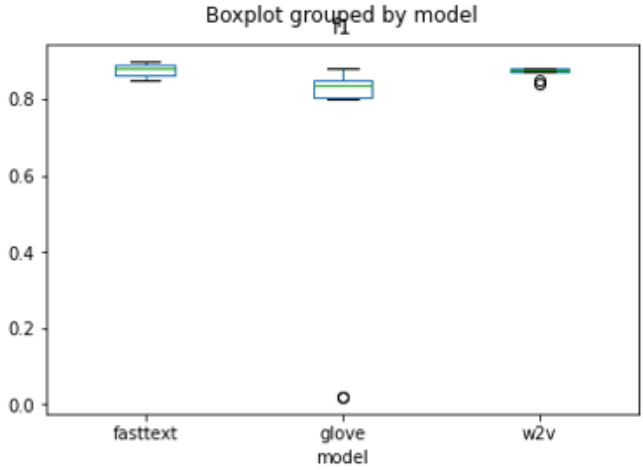
*(b) Results obtained in this paper for 100 epochs as initial run for base architecture*

| Embedded matrix | Average | | |
|---|---|---|---|
| | Recall | Precision | F1 |
| Glove | 0.82 | 0.79 | 0.79 |
| Word2vec | 0.88 | 0.90 | 0.88 |
| FastText | 0.83 | 0.85 | 0.83 |

To begin the ANOVA calculation, the average F1 metric results of each display obtained from cross-validated training for Word2vec, Glove, and FastText were collected. An analysis of the distribution of said data was carried out, where it can be seen with the naked eye in Fig. 1a that for the model with Glove atypical values are reflected, for example, containing an average of 0.02 (see Fig. 1a) for *fold* number 6 and 10, on the other hand, Word2vec presents outliers to the average, in contrast to FastText which contains no outliers.

Now, as already mentioned, the purpose of this proposal was to evaluate the difference between each model and the proposals implemented with respect to the metric *F1*. The overall mean of the metric

*F1*, as shown in Fig. 2a, was 0.81 for 30 samples (N) and its confidence interval (*CI*, for its acronym in English) at 95% was (0.72,0.89). The specific means for the groups by model with *N=10* were the following (see Fig. 2b): FastText with a mean of 0.88 and a CI at 95% of (0.86,0.88), for Glove's case obtained an average of 0.67 with a 95% CI of (0.45,0.90), on the other hand, Word2vec returned an average of 0.87 and a 95% CI of (0.86,0.88).




*(a) Boxplot of the averages of the F1 metrics belonging to the models trained with the 3 embedded matrices*

| | kf1 | kf2 | kf3 | kf4 | kf5 | kf6 | kf7 | kf8 | kf9 | kf10 |
|---|---|---|---|---|---|---|---|---|---|---|
| W2vec | 0.88 | 0.88 | 0.87 | 0.88 | 0.88 | **0.85** | 0.87 | 0.88 | 0.87 | **0.84** |
| Glove | 0.80 | 0.88 | 0.85 | 0.81 | 0.86 | **0.02** | 0.85 | 0.83 | 0.84 | **0.02** |
| FastText | **0.90** | 0.89 | 0.89 | 0.85 | 0.87 | 0.86 | 0.85 | 0.88 | 0.88 | **0.90** |

*(b) Table of averages of the F1 metric for models trained with pre-trained matrices*

*Fig. 1: Table of averages of the F1 metric for the trained models and distribution diagram of said data*

| Variable | N | Average | SD | SE | 95% Conf | Interval |
|---|---|---|---|---|---|---|
| **F1** | 30 | 0.81 | 0.21 | 0.04 | 0.72 | 0.89 |

*(a) Summary table for total samples against F1 metric averages*

| Variable | Average | SD | SE | 95% Conf | Interval |
|---|---|---|---|---|---|
| **V1** | 0.57 | 0.05 | 0.01 | 0.54 | 0.60 |
| **V2** | 0.70 | 0.04 | 0.01 | 0.67 | 0.72 |
| **V3** | 0.88 | 0.02 | 0.01 | 0.86 | 0.89 |

*(b) Table of F1 averages of individual models*

*Fig. 2: Tables of general and particular averages of the models analyzed with respect to F1*

The standard significance value of = 0.05 was taken as a reference, however, when calculating ANOVA for the samples presented, it can be seen in Table 6 that there is a statistically significant difference between the models presented because it was obtained a p=1.148591e-16. Therefore, with the observed data, there is sufficient evidence to assume a significant difference between the models exposed in this evaluation. It is worth mentioning that the assumptions of the test were verified using the Kruskall-Wallis non-parametric test. In addition, the honestly significant difference test of *Tukey (Tukey's HSD)* was performed, which is used to test the differences between the means of the sample in terms of significance, testing the differences by peers.

*Table 6: ANOVA result for the models with respect to F1*

| | degrees of freedom | Sum of Squares | average of squares | F | P |
|---|---|---|---|---|---|
| **Models** | 2.0 | 0.48 | 0.24 | 191.17 | 1.14e-16 |
| **Residual** | 27.0 | 0.03 | 0.001 | | |

## *4. Conclusions and future work*

NLP, a subset of AI, has been great allies in solving text classification problems. However, the solution to these problems is generally reserved for large-scale problems with large volumes of data samples, so working with databases with few examples suggests low results with respect to classification performance. This problem has been present when trying to optimize the classification of software requirements since during the analysis of these they are embodied in a document that is frequently represented by sentences with little text or information. Furthermore, requirements data sets only contain hundreds to thousands of documents, which is orders of magnitude less in volume than is typically considered necessary for deep learning. In addition, taking into account that there are different classes of requirements, especially the NFRs that are obtained from the Promise data set, which makes the classification task difficult to obtain desired results. That is why in this work, an investigation was carried out to determine which were the best strategies used in the state of the art that led to obtaining acceptable results in each of the investigations, in order to unify them and observe if said strategies together reflected a performance efficiency for the classification of NFRs using convolutional neural networks. In principle, it was possible to observe, for experimentation, how to vectorize the data set with the embedding of words, apply methods of Random over sampling strategies, hyper parameterize the configuration when performing the data partition, an increase in the average of the metrics was reflected Recall, Precision and F1 against the state of the art, since unlike [10] up to 30% was obtained in the average increase of the mentioned metrics. This is the first guideline to apply them to CNN. When implementing these proposals to the CNN architecture, as well as performing its hyper parameterization, it was decided to test with an embedded layer with or without weights, thus showing the importance of using pre-trained matrices that allow improvement in terms of the classification of text. To determine if there really was an improvement in the classifier, the ANOVA analysis was performed, which revealed a p-value of 0.05, therefore, according to the standard significance of, if there is a significant improvement between the models presented. Hence, it can be said that the application of the proposals for the classification of NFR with CNN resulted in the improvement of the performance of the classifier with respect to the state of the art.

The future work that is planned to be carried out in the first instance is to search for data sets with more examples to observe the performance behavior of the classifier, as well as to experiment with recurrent neural networks such as LSTM. Also, due to the challenge represented by carrying out the training with k-folds and the CNN base architecture on hardware in which the memory did not support the execution, it is planned to look for an institution that can provide some resources for research on this topic.

## *References*

[1]. José Alfonso Aguilar, Anibal Zaldívar-Colado, Carolina Tripp-Barba, Roberto Espinosa, Sanjay Misra, Carlos Eduardo Zurita: A Survey About the Impact of Requirements Engineering Practice in Small-Sized Software Factories in Sinaloa, Mexico. International Conference on Computational Science and Its Applications, 331–340. Springer, 2018.

[2]. Alla Sujatha, Pilar Pazos, Rolando DelAguila: The Impact of Requirements Management Documentation on Software Project Outcomes in Health Care. IIE Annual Conference. Proceedings, 1419–1423. Institute of Industrial and Systems Engineers (IISE), 2017.

[3]. S. Almeyda, A. Davila: Process Improvement in Software Requirements Engineering: A Systematic Mapping Study. Programming and Computer Software, 48(8):513–533, 2022.
[4]. Aybuke Aurum, Claes Wohlin: A value- based approach in requirements engineering: explaining some of the fundamental concepts. International Working Conference on Requirements Engineering: Foundation for Software Quality, 109–115. Springer, 2007.
[5]. Christoph Becker, Stefanie Betz, Ruzanna Chitchyan, Leticia Duboc, Steve M. Easterbrook, Birgit Penzenstadler, Norbet Seyff, Colin C. Venters: Requirements: The key to sustainability. IEEE Software, vol.33:56–65, 2016.
[6]. Agustin Casamayor, Daniela Godoy, Marcelo R. Campo: Identification of non-functional requirements in textual specifications: A semi-supervised learning approach. Information and Software Technology, vol.52:436–445, 2010.
[7]. Lawrence Chung, Brian A. Nixon, Eric Yu, John Mylopoulos: Non-functional requirements in software engineering. Springer Science & Business Media, 2012.
[8]. J. R. Aguilar Cisneros, C. A. Fernandez-y-Fernandez, Genaro de la Rosa Garcia, A. Leon: Automotive Post-Collision Control Software System: Requirements and Verification. Programming and Computer Software, 47(8):735–745, 2021.
[9]. F. Dalpiaz, A. Ferrari, X. Franch, C. Palomares: Natural Language Processing for Requirements Engineering: The Best Is Yet to Come. IEEE Software, vol.35:115–119, 2018, ISSN 0740-7459.
[10]. Vivian Lin Fong: Software Requirements Classification Using Word Embeddings and Convolutional Neural Networks. Cal Poly, 2018.
[11]. Martin Glinz: On non-functional requirements. Requirements Engineering Conference, 2007. RE'07. 15th IEEE International, 21–26. IEEE, 2007.
[12]. Margaret Hamill, Katerina Goseva-Popstojanova: Common trends in software fault and failure data. IEEE Transactions on Software Engineering, vol.35:484– 496, 2009.
[13]. Ishrar Hussain, Olga Ormandjieva, Leila Kosseim: LASR: A tool for large scale annotation of software requirements. 2012 Second IEEE International Workshop on Empirical Requirements Engineering (EmpiRE), 57–60. IEEE, 2012.
[14]. IEEE Computer Society, Software Engineering Standards Committee, IEEE SA Standards Board: IEEE Recommended Practice for Software Requirements Specifications. 1998.
[15]. Raul Navarro Almanza, Reyes Juárez-Ramirez, Guillermo Licea: Towards Supporting Software Engineering Using Deep Learning: A Case of Software Requirements Classification. 2017 5th International Conference in Software Engineering Research and Innovation (CONISOFT), 116–120. IEEE, 2017.
[16]. Mohamad Kassab: Non-functional requirements: modeling and assessment. VDM Verlag, 2009.
[17]. Marjo Kauppinen, Juha Savolainen, Tomi Mannisto: Requirements engineering as a driver for innovations. 15th IEEE International Requirements Engineering Conference (RE 2007), 15–20. IEEE, 2007.
[18]. Youngjoong Ko, Sooyong Park, Jungyun Seo, Soonhwang Choi: Using classification techniques for informal requirements in the requirements analysis-supporting system. Information and Software Technology, vol.49:1128–1140, 2007.
[19]. Zijad Kurtanovic, Walid Maalej: Automatically classifying functional and non-functional requirements using supervised machine learning. Requirements Engineering Conference (RE), 2017 IEEE 25th International, 490–495. IEEE, 2017.
[20]. Timo O. A. Lehtinen, Mika Mäntylä, Jari Vanhanen, Juha Itkonen, Casper Lassenius: Perceived causes of software project failures–An analysis of their relationships. Information and Software Technology, vol.56:623–643, 2014.
[21]. Tomas Mikolov: Statistical language models based on neural networks. Presentation at Google, Mountain View, 2nd April, vol.80, 2012.
[22]. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, Jeffrey Dean: Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems, 3111–3119, 2013.
[23]. Nan Niu, Sjaak Brinkkemper, Xavier Franch, Jari Partanen, Juha Savolainen: Requirements engineering and continuous deployment. IEEE software, vol.35(2):86–90, 2018.
[24]. Carla Pacheco, Ivan Garcia, Miryam Reyes: Requirements elicitation techniques: a systematic literature review based on the maturity of the techniques. IET Software, vol.12:365–378, 2018.

[25]. J. Manuel Pérez-Verdejo, Angel Juan Sanchez Garcia, Jorge Octavio Ocharán-Hernández, Efrén Mezura-Montes, Karen Cortés-Verdin: Requirements and GitHub Issues: An Automated Approach for Quality Requirements Classification. Programming and Computer Software, 47:704–721, 2021.
[26]. Abderahman Rashwan, Olga Ormandjieva, Rene Witte: Ontology-based classification of non-functional requirements in software specifications: a new corpus and SVM-based classifier. Computer Software and Applications Conference (COMPSAC), 2013 IEEE 37th Annual, 381–386. IEEE, 2013.
[27]. Patrick Rempel, Patrick Máder: Preventing defects: The impact of requirements traceability completeness on software quality. IEEE Transactions on Software Engineering, vol.43:777–797, 2017.
[28]. Kevin Ryan: The role of natural language in requirements engineering. Proceedings of the IEEE International Symposium on Requirements Engineering, 240– 242. IEEE, 1993.
[29]. J. Sayyad Shirabad, Tim J. Menzies: The PROMISE Repository of Software Engineering Databases, 2005. http://promise.site.uottawa.ca/SERepository.
[30]. Belal Shanyour, Abdallah Qusef: Global Software Development and its Impact on Software Quality. 2018 Fifth International Symposium on Innovation in Information and Communication Technology (ISIICT), 1–6. IEEE, 2018.
[31]. Ian Sommerville: Software engineering, 9th Edition. Pearson, 2011.
[32]. Jason Van Hulse, Taghi M. Khoshgoftaar, Amri Napolitano: Experimental perspectives on learning from imbalanced data. Proceedings of the 24th international conference on Machine learning, 935–942, 2007.
[33]. Jonas Winkler, Andreas Vogelsang: Automatic classification of requirements based on convolutional neural networks. 2016 IEEE 24th International Requirements Engineering Conference Workshops (REW), 39–45. IEEE, 2016.

## *Информация об авторах / Information about authors*

Сандра Эстефания МАРТИНЕС ГАРСИЯ имеет магистерскую степень Технологического университета Миштека (2021 год) в области прикладных вычислительных технологий и Технологического института Оахака в области вычислительной техники (2019 год). В настоящее время она является специалистом по управлению данными и техническим руководителем мексиканского отделения компании Bosch. Ее области интересов включают искусственный интеллект, разработку программного обеспечения, управление разработкой требований, автоматизацию процессов, бизнес-аналитику, аналитику данных и науку о данных.

Sandra Estefanía MARTINEZ GARCIA obtained a master's degree in Applied Computing Technologies from the Universidad Tecnológica de la Mixteca in 2021 and Computer engineering from Instituto Tecnológico Del Valle de Oaxaca in 2019. She is currently a data scientist and technical lead at Bosch Mexico. Her areas of interest include Artificial Intelligence, Software Engineering, Requirements Engineering, Process Automation, Business Analytics, Data Analytics and Data Science.

Карлос Альберто ФЕРНАНДЕС-И-ФЕРНАНДЕС имеет степень PhD университета Шеффилда по программированию, эксперт в области программирования. В настоящее время возглавляет Институт вычислений в Технологическом университете в мексиканском регионе Миштека, координирует магистерские программы по прикладным аспектам вычислительных технологий. Сфера научных интересов: визуальное моделирование, гибкие технологии разработки и формальные спецификации программного обеспечения.

Carlos Alberto FERNÁNDEZ-Y-FERNÁNDEZ – Software Engineering expert with a Ph.D from the University of Sheffield. He currently leads the Institute of Computing at Universidad Tecnológica de la Mixteca and coordinates the Master's program in Applied Computing Technologies. His research interests include visual modeling, agile methods, and formal software specification.

Эрик Г. РАМОС ПЕРЕС – Профессор-исследователь Технологического университета Миштека. Имеет степени магистра Технологическом университете Ла-Микстека в области прикладных вычислительных технологий (2016 год) и по вычислительной технике (2001 год).

Является координатором Виртуального университета. Его области научных интересов включают машинное обучение и автономную навигацию беспилотных аппаратов.

Erik G. RAMOS PÉREZ – Professor-Researcher at the Technological University of La Mixteca. He obtained a master's degree in Applied Computing Technologies and Computer Engineering from the Technological University of La Mixteca in 2016 and 2001 respectively. He is coordinator of the Virtual University. His areas of research interest include Machine Learning and Autonomous Navigation with drones.