Evaluating Non-Functional Requirements Classification for Spanish Text: Traditional vs. Deep Learning Approaches

María Isabel Limaylla-Lunarejo
Database Lab, Fac. Informática
Universidade da Coruña
15071 A Coruña, Spain
`maria.limaylla@@udc.es`

Nelly Condori-Fernandez
Centro Singular de Investigación en Tecnoloxías
Universidad de Santiago de Compostela
15782 Santiago de Compostela, Spain
`n.condori.fernandez@usc.es`

Miguel R. Luaces
Database Lab, Fac. Informática
Universidade da Coruña
15071 A Coruña, Spain
`miguel.luaces@udc.es`

Abstract

The automatic classification of non-functional requirements helps to reduce time and effort for the stakeholders. Several techniques have been used for this task, including the latest techniques in Machine Learning (ML) and Natural Language Processing (NLP), such as pre-trained models, with promising results. This research aims to analyze the performance metrics to classify requirements into sub-classes of non-functional type. Six distinct algorithms, including both traditional machine learning (ML) and deep learning (DL), were trained using a Spanish-translated PROMISE NFR dataset to assess and compare their performance outcomes. The findings reveal that Support Vector Machine (SVM) with Bag of Word technique, and fastText overperformed the other algorithms. However, fastText stands out between the two due the ease of implementation and the absence of data pre-processing.

1 Introduction

Non-functional requirements (NFR) significantly contribute to the overall quality of software. The automatic classification of NFR offers several benefits, including a reduction in stakeholder effort, decreased stakeholder involvement, real-time classification, and simplified identification of stakeholders with expertise, among others. The automatic classification of non-functional requirements has been extensively studied Binkhonain and Zhao [2019], Tóth and Vidács [2018b,a], Haque et al. [2019], Khatian et al. [2021]. As requirements are expressed in natural language, several NLP techniques have been used in this process in the last years with promising results. However, there is a lack of analysis for non-functional requirements in others languages. Analyzing requirements in the language they were originally written contributes to the quality of the requirements. In the most recent analysis published by 'Instituto Cervantes' for 2022, the Spanish language ranks as the second most widely spoken native language, representing around 6.3% of the global population Instituto Cervantes [2022]. Given the importance of the Spanish language, it's important to direct our focus on this language in the field of requirements engineering (RE).

In recent years, there has been an accelerated development concerning Deep Learning (DL) for NLP (e.g., large language models), which makes it necessary to continue the evaluation and comparison of the new models with traditional ones. We will refer to the traditional algorithm as 'shallow ML algorithm', encompassing simple neural networks and algorithms such as Naive Bayes (NB) or Support Vector Machine (SVM), according to Janiesch et al. [2021]. Prior to the rise of neural networks for NLP, shallow ML algorithms in combination with NLP techniques (such as tokenization, stemming, lemmatization, text vectorization) obtained good results on classification, with SVM as the most popular Binkhonain and Zhao [2019]. In addition to that, the use of n-grams could have an impact on performance results when combined with other NLP techniques. Zheng [2019]. Since 2017 with the appearance of the transformers architecture (based on neural networks) Vaswani et al. [2017] and the transfer learning application in NLP, pre-trained models has appears with positive findings. Some popular models are based on Bert Devlin et al. [2018] (with billions of parameters) and trained with large corpus in several languages.

In this research, we compare several models based on Machine Learning (ML) algorithms and NLP techniques for automatic non-functional requirements classification to find the best combinations, focus on requirements written in Spanish. The main contributions of this paper are:

- A quantitative comparison of performance metrics to compare the shallow and DL algorithms for NFR classification.
- The use of pre-trained models (e.g., BETO) on NFR classification, trained with Spanish corpus.
- This paper also serves as a tutorial for the classification of non-functional requirements written in Spanish.

2 Related Work

Several studies haven been used ML algorithms to improve the automatic classification of NFR Rahman et al. [2019], Tóth and Vidács [2018a], Baker et al. [2019], Haque et al. [2019], Khatian et al. [2021], Dias Canedo and Cordeiro Mendes [2020]. Binkhonain and Zhao (2019) provide a comprehensive review of 24 articles that employ ML algorithms to identify non-functional requirements Binkhonain and Zhao [2019]. López-Hernández et al. (2021) also performed a more recent review of requirements classifications, analysing 14 studies between 2016-2020. They found that PROMISE NFR is the database most used, NB and SVM as the most used algorithms for shallow ML and Convolutional neural network (CNN) for neural network and half of the studies focused only in NFR classification López-Hernández et al. [2021]. In Iipinge and Hu [2023], challenges such as lack of training and evaluation data are outlined, and recommendations are provided, such as the utilization of both ML and NLP, to analyze security features Iipinge and Hu [2023].

Few studies of requirements in different languages of English have been found. Arabic language Alsawareah et al. [2023] and Chinese language Alsawareah et al. [2023] are examples of studies on others languages. Regarding to requirements written in Spanish, in our previous research we translated the PROMISE dataset and explored the classification of requirements in functional and non-functional, finding that the shallow ML algorithms outperform DL algorithms. In this research, we used the translated PROMISE dataset to train several algorithms to analyse the automatic classification of NFRs in the Spanish language.

3 Research Design

3.1 Research Questions

- **RQ1**: How do different combinations of text vectorization techniques and ML algorithms impact the performance of classifying non-functional requirements in sub-classes?
- **RQ2**: How does the use of Bigram/Trigram in text vectorization affect the classification results in Shallow ML algorithms?
- **RQ3**: Does the DL models overperform the Shallow ML algorithms?

3.2 Dataset

The PROMISE NFR is one of the 89 datasets of the PROMISE repository Sayyad Shirabad and Menzies [2005], a repository widely used by the software engineering community. This dataset is labeled into 12 classes, where one is the 'Functional' class, and the other 11 are non-functional requirements. In this research, we used only those labeled as sub-classes of non-functional requirements: Functional, Availability, Legal, Look and feel, Maintainability, Operational, Performance, Scalability, Security, Usability, Fault tolerance, and Portability, making a total of 370 requirements. Table 1 shows the quantity of records per label.

Regarding the non-functional, there are some unbalanced classes and low representative of at least one class: Portability, with only one requirement. In Limaylla-Lunarejo et al. [2023], a translation of this dataset was performed to obtain a dataset in the Spanish language, a 'Translated PROMISE'. We obtained this dataset from Zenodo[1], when it was published for public use. For this research, we decided to remove the classes whose number requirements do not exceed a threshold (20 requirements): Legal, Maintainability, Fault tolerance, and Portability. Finally, 329 requirements for Translated PROMISE were processed, divided into 280 and 49 for training and testing, respectively. Due the small size of the dataset, we decide to use a split of 85% train and 15% test.

Table 1: Summary of records - PROMISE NFR

Label	Quantity PROMISE NFR
PO = Portability	1
FT = Fault tolerance	10
L = Legal	13
MN = Maintainability	17
A = Availability	21
SC = Scalability	21
LF = Look and feel	38
PE = Performance	54
O = Operational	62
SE = Security	66
US = Usability	67
Total	370

3.3 Classification Process

We performed an experiment to evaluate and compare the performance of several algorithms, based on the strategy proposed by Dalal and Zaveri (2011) Dalal and Zaveri [2011]. For Shallow ML algorithms, we used traditional NLP techniques, like tokenization, stopwords, and stemming for pre-processing; Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF) for extract features (text vectorization); and Multinomial Naive Bayes (MNB), Multinomial Logistic Regression (MLR), and Support Vector Machine (SVM) as selected algorithms. According to some research (Zheng [2019], Maalej et al. [2016]), the use of n-grams leads to better performance scores in text classification in some cases. We applied the combination of Unigram, Bigram, and Trigram with BoW and with TF-IDF, resulting in six combinations for every Shallow ML algorithm (18 in total) to validate the impact of the performance for the n-grams technique. Regarding the DL algorithms, no data pre-processing is required for the selected algorithms: the Convolutional Neural Network (CNN), BETO Cañete et al. [2020] (a Bert-based model pre-trained on several Spanish corpora), and fastText Joulin et al. [2016]. Each one has its own technique for text vectorization (e.g., word embedding).

The process followed in this study is shown in Figure 1. The training phase was performed on each algorithm and tuning the hyperparameter associated on the training dataset. We present the best hyperparameters for each algorithm in AppendixA. Once the best hyperparameters have been

[1]https://doi.org/10.5281/zenodo.7311148

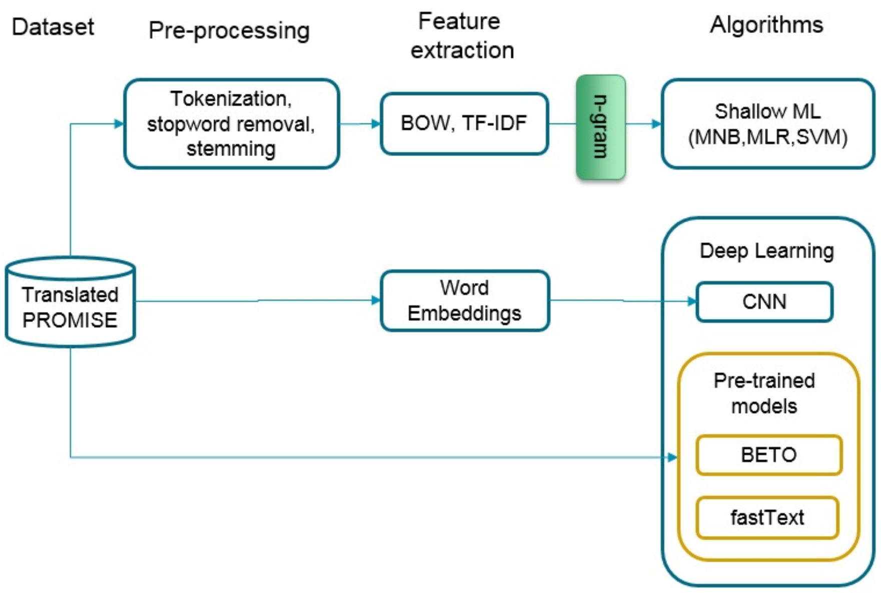

Figure 1: Experiment steps for requirement classification

obtained, all the models were tested in the same test split and compare using well-known metrics, like accuracy, precision, recall and f1-score (weigthed).

4 Results

This section presents the performance metrics for NFR classification training with Translated PROMISE dataset (RQ1/R3) and using Bigram/Trigram in combination with the text vectorization techniques for the Shallow ML algorithms (RQ2).

4.1 NFR Classification using Translated PROMISE (RQ1/RQ3)

We trained the Translated PROMISE NFR dataset with all the combinations of algorithms and text vectorization to obtain the best requirement classification performance. Table 2 shows the resulting accuracy and weighted average of precision, recall, and f1-score metrics for the classification processes. The values in bold indicate the highest weighted f1-scores.

Table 2: Performance Classification (RQ1)

Algorithm	NLP technique	accuracy	weighted avg		
			precision	recall	f1-score
MNB	BoW	0.69	0.71	0.69	0.69
	TF-IDF	0.67	0.69	0.67	0.67
MLR	BoW	0.73	0.73	0.73	0.72
	TF-IDF	0.67	0.68	0.67	0.67
SVM	BoW	0.76	0.78	0.76	**0.75**
	TF-IDF	0.73	0.76	0.73	0.73
CNN		0.73	0.76	0.73	0.72
BETO		0.69	0.81	0.69	0.67
FASTTEXT		0.76	0.76	0.76	**0.75**

SVM in combination with BoW and FastText achieves the best weighted f1-score for all the models: 0.75. Both models performed the training process in less than 10 minutes, however, the FastText model was easier to implement and since it manages its own tokenizer internally, it was not necessary to perform any pre-processing technique. SVM in combination with TF-IDF, CNN, and MLR with BoW also presented f1-score values above 0.7. Regarding weighted precision and recall, precision is slightly higher than recall, except for BETO, who has a higher value for precision (0.81).

4.2 NFR Classification using Translated PROMISE and applying Bigram/Trigram (RQ2)

To identify if the use of Bigram or Trigram has an impact on the performance classification for the Shallow ML algorithms, we combined the BoW and TF-IDF with the use of n-gram for the Shallow ML algorithms. The CountVectorizer and TfidfVectorize classes, provided by the scikit-learn library, allow the configuration of a range of n-grams considered in the text vectorization process. We considered a range of (1, 2), which means unigrams and bigrams and presented here as Bigrams, and a range of (1,3), which means unigrams, bigrams, and trigrams, presented as Trigrams. Table 3 shows the results using both cases.

Table 3: Performance Classification with Bigram/Trigram (RQ2)

Algorithm	NLP technique	accuracy	weighted avg		
			precision	recall	f1-score
MNB	BoW Bigram	0.71	0.72	0.71	0.71
	BoW Trigram	0.71	0.72	0.71	0.71
	TF-IDF Bigram	0.69	0.70	0.69	0.69
	TF-IDF Trigram	0.67	0.68	0.67	0.67
MLR	BoW Bigram	0.73	0.74	0.73	0.72
	BoW Trigram	0.71	0.73	0.71	0.70
	TF-IDF Bigram	0.71	0.74	0.71	0.70
	TF-IDF Trigram	0.73	0.77	0.73	0.72
SVM	BoW Bigram	0.73	0.73	0.73	0.73
	BoW Trigram	0.76	0.76	0.76	0.74
	TF-IDF Bigram	0.71	0.75	0.71	0.70
	TF-IDF Trigram	0.73	0.76	0.73	0.72

A comparison of these results with the results of the previous subsection (RQ1) demonstrates that depends on the algorithm and the text vectorization technique if the Bigram or Trigram improves the f1-score metrics. For MNB in combination with BoW, the use of Bigram and Trigram obtained better results than Unigram (over 0.02), while with TF-IDF only the use of Bigram improve the results (0.02). For MLR in combination with BoW, Bigram gets the same results and Trigram even lower values, while with TF-IDF Bigram gets better results in 0.03 and Trigram in 0.05. Finally, for SVM in combination with BoW and with TF-IDF, Bigram, and Trigram get lower values than Unigram.

5 Discussion

The use of neural networks is becoming more common, including in requirements classification López-Hernández et al. [2021]. We can observe, based on the initial result, that both shallow ML and DL achieved similar outcomes. Regarding RQ1, SVM with BoW and fastText get the best performance. We found that SVM obtained better results with BoW technique, which is contrary to the results obtained in Limaylla-Lunarejo et al. [2023], where SVM achieved better performance using TF-IDF in FR/NFR classification.This might be attributed to the fact that non-functional requirements often share numerous common words and cardinal numbers, causing certain words that could be important for classifying a specific subclass to be undervalued when employing TF-IDF. Nevertheless, the literature presents contrasting findings. In Dias Canedo and Cordeiro Mendes [2020], the combination of SVM with TF-IDF outperformed the results when compared to using BoW with the PROMISE dataset. However, in Haque et al. [2019], the situation is reversed with an expanded version of PROMISE dataset.

One of the research questions aimed to explore the influence of employing n-gram within different algorithm and feature extraction combinations (RQ2). In our previous research we introduced several binary classifiers (FR/NFR), with most of them demonstrating improvement when employing bigrams and unigrams Limaylla-Lunarejo et al. [2023]. On the other hand, Suhaidi et al. (2021) discuss the effect of using n-grams in the text pre-processing method, mentioning that this technique can make a classifier more restrictive Suhaidi et al. [2021]. According to the results of the Experiment Two, the MNB algorithm is the one that most reflects the impact of the use of n-grams, with an improvement with both BoW and TF-IDF techniques. MLR with TF-IDF also presents an improvement with bigrams, and SVM does not show any improvement. In the analysis of sub-classes of NFRs, we are dealing with multi-class classification, and with a text that include more technical and numerical vocabulary, that can lead to limited improvement when using n-grams.

Regarding RQ3, we consider that even though fastText and SVM obtained similar results, fastText is a highly potent algorithm. Its ease of implementation, automated tuning process, and the absence of data pre-processing make it the optimal choice in terms of the trade-off between implementation time and performance. Similar results were found in Tiun et al. [2020], where fastText was the best model for binary requirements classification (functional/non-functional) and in our previous research Limaylla-Lunarejo et al. [2023]. However, just like other DL algorithms, it is considered a 'black box,' and one of its main limitations is its interpretability in text classification Kowsari et al. [2019].

6 Conclusions

In this research, we have presented a comparison between some traditional ML algorithms in combination with text vectorizations, and some DL algorrithms to classify automatically sub-classes of NFR written in Spanish. Our findings reveal that SVM with BoW and fastText overperformed the other algorithms, with a weighted f1-score of 0.75. Nevertheless, fastText stands out due to its speed and ease of implementation. The results also reveal that, although a general rule for the use of n-gram in combination with shallow algorithms and TF-IDF techniques cannot be confirmed, its use in the training phase is recommended, since it can improve some results in certain cases.

About future work, our plan includes replicating the experiment with the extended PROMISE NFR dataset Lima et al. [2019], a new dataset that expands the original PROMISE NFR. We will also explore strategies to balance the dataset in order to enhance performance and include all the sub-classes of non-functional category.

References

Bayan Alsawareah, Ahmad Althunibat, and Bilal Hawashin. Classification of arabic software requirements using machine learning techniques. In *2023 International Conference on Information Technology (ICIT)*, pages 631–636. IEEE, 2023.

Cody Baker, Lin Deng, Suranjan Chakraborty, and Josh Dehlinger. Automatic multi-class non-functional software requirements classification using neural networks. In *2019 IEEE 43rd annual computer software and applications conference (COMPSAC)*, volume 2, pages 610–615. IEEE, 2019.

Manal Binkhonain and Liping Zhao. A review of machine learning algorithms for identification and classification of non-functional requirements. *Expert Syst. Appl.: X*, 1:100001, 2019.

José Cañete, Gabriel Chaperon, Rodrigo Fuentes, Jou-Hui Ho, Hojin Kang, and Jorge Pérez. Spanish pre-trained bert model and evaluation data. In *PML4DC at ICLR 2020*, 2020.

Mita K Dalal and Mukesh A Zaveri. Automatic text classification: a technical review. *International Journal of Computer Applications*, 28(2):37–40, 2011.

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

Edna Dias Canedo and Bruno Cordeiro Mendes. Software requirements classification using machine learning algorithms. *Entropy*, 22(9):1057, 2020.

Md Ariful Haque, Md Abdur Rahman, and Md Saeed Siddik. 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)*, pages 1–5. IEEE, 2019.

Vilho James Iipinge and Mengting Hu. Automated methodology for classifying non-functional requirements: Challenges, validation aspects & new research directions. In *Proceedings of the 2023 9th International Conference on Computing and Artificial Intelligence*, pages 568–575, 2023.

Instituto Cervantes. El español una lengua viva, 2022. URL `https://cvc.cervantes.es/lengua/espanol_lengua_viva/`. [Online; accessed 28-August-2023].

Christian Janiesch, Patrick Zschech, and Kai Heinrich. Machine learning and deep learning. *Electronic Markets*, 31(3):685–695, 2021.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

Vajeeha Mir Khatian, Qasim Ali Arain, Mamdouh Alenezi, Muhammad Owais Raza, Fariha Shaikh, and Isma Farah. Comparative analysis for predicting non-functional requirements using supervised machine learning. In *2021 1st International Conference on Artificial Intelligence and Data Analytics (CAIDA)*, pages 7–12. IEEE, 2021.

Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, and Donald Brown. Text classification algorithms: A survey. *Information*, 10(4):150, 2019.

Márcia Lima, Victor Valle, Estevão Costa, Fylype Lira, and Bruno Gadelha. Software engineering repositories: expanding the promise database. In *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, pages 427–436, 2019.

María-Isabel Limaylla-Lunarejo, Nelly Condori-Fernandez, and Miguel R Luaces. Requirements classification using fasttext and beto in spanish documents. In *Requirements Engineering: Foundation for Software Quality: 29th International Working Conference, REFSQ 2023, Barcelona, Spain, April 17–20, 2023, Proceedings*, pages 159–176. Springer, 2023.

Delmer Alejandro López-Hernández, Jorge Octavio Ocharán-Hernández, Efrén Mezura-Montes, and Ángel J Sánchez-García. Automatic classification of software requirements using artificial neural networks: A systematic literature review. In *2021 9th International Conference in Software Engineering Research and Innovation (CONISOFT)*, pages 152–160. IEEE, 2021.

Walid Maalej, Zijad Kurtanović, Hadeer Nabil, and Christoph Stanik. On the automatic classification of app reviews. *Requirements Engineering*, 21(3):311–331, 2016.

Md Abdur Rahman, Md Ariful Haque, Md Nurul Ahad Tawhid, and Md Saeed Siddik. Classifying non-functional requirements using rnn variants for quality software development. In *Proceedings of the 3rd ACM SIGSOFT International Workshop on Machine Learning Techniques for Software Quality Evaluation*, pages 25–30, 2019.

J. Sayyad Shirabad and T.J. Menzies. The PROMISE Repository of Software Engineering Databases. School of Information Technology and Engineering, University of Ottawa, Canada, 2005. URL `http://promise.site.uottawa.ca/SERepository`.

Mustazzihim Suhaidi, Rabiah Abdul Kadir, and Sabrina Tiun. A review of feature extraction methods on machine learning. *dimension*, 6(22):51–59, 2021.

S Tiun, UA Mokhtar, SH Bakar, and S Saad. Classification of functional and non-functional requirement in software requirement using word2vec and fast text. In *journal of Physics: conference series*, volume 1529, page 042077. IOP Publishing, 2020.

László Tóth and László Vidács. Study of various classifiers for identification and classification of non-functional requirements. In *International Conference on Computational Science and Its Applications*, pages 492–503. Springer, 2018a.

László Tóth and László Vidács. Study of various classifiers for identification and classification of non-functional requirements. In *Computational Science and Its Applications–ICCSA 2018: 18th International Conference, Melbourne, VIC, Australia, July 2-5, 2018, Proceedings, Part V 18*, pages 492–503. Springer, 2018b.

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

Yuhan Zheng. An exploration on text classification with classical machine learning algorithm. In *2019 International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI)*, pages 81–85. IEEE, 2019.

A Best Hyperparameters

Table 4 and Table 5 show the best hyperparameters used for the shallow ML and DL algorithms, respectively. Since fastText is an algorithm with its own tuning process, there is no need to specify or obtain the hyperparameters.

Table 4: Best Hyperparameters - Shallow ML algorithms

Alg.	**Hp**	**BoW Unigram**	**BoW Bigram**	**BoW Trigram**	**TF-IDF Unigram**	**TF-IDF Bigram**	**TF-IDF Trigram**
MNB	alpha:	0.01	0.01	0.01	0.01	0.01	0.1
	fit_prior:	false	true	false	true	true	true
MLR	C:	2	10	1	10	10	100
	solver:	newton-cg	newton-cg	newton-cg	newton-cg	newton-cg	newton-cg
SVM	C:	10	10	10	10	10	1.0
	gamma:	0.05	0.05	0.01	0.5	0.05	0.001
	kernel:	rbf	linear	rbf	rbf	rbf	linear

Table 5: Best Hyperparameters - DL algorithms

Algorithms	**Hyperparameters**
CNN	batch_size: 16 epochs: 40 filtersmap: 100 learning_rate: 0.001
BETO	batch_size: 32 epochs: 10 learning_rate: 0.00005