IPTEK The Journal of Technology and Science, 32(3), 2021 (e/pISSN:2088-2033, 0853-4098)

DOI: 10.12962/j20882033.v32i3.13005 | Received 12 September, 2021; Accepted 12 Dec, 2021

ORIGINAL RESEARCH

A SYSTEMATIC COMPARISON OF SOFTWARE REQUIREMENTS CLASSIFICATION

Fajar Baskoro*[1] | Rasi Aziizah Andrahsmara[2] | Brian Rizqi Paradisiaca Darnoto[2] | Yoga Ari Tofan[2]

[1]Dept. of Informatics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia

[2]Informatics Master Programme, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia

Correspondence

*Fajar Baskoro, Dept of Information Informatics, Institut Teknologi Sepuluh Nopember, Surabbaya, Indonesia. Email: fajar@if.its.ac.id

Present Address

Gedung Teknik Informatika, Jl. Teknik Kimia, Kampus ITS Sukolilo, Surabaya 60111, Indonesia

Abstract

Software requirements specification (SRS) is an essential part of software development. SRS has two features: functional requirements (FR) and non-functional requirements (NFR). Functional requirements define the needs that are directly in contact with stakeholders. Non-functional requirements describe how the software provides the means to carry out functional requirements. Non-functional requirements are often mixed with functional requirements. This study compares four primarily used machine learning methods for classifying functional and non-functional requirements. The contribution of our research is to use the PROMISE and SecReq (ePurse) dataset, then classify them by comparing the FastText+SVM, FastText+CNN, SVM, and CNN classification methods. CNN outperformed other methods on both datasets. The accuracy obtained by CNN on the PROMISE dataset is 99% and on the Seqreq dataset is 94%.

KEYWORDS:

CNN, FaxtText,Requirements Classification, Software Requirements, SVM

1 | INTRODUCTION

Software requirement specification (SRS) is the most critical step during the software development cycle. SRS is divided into functional requirements (FR) and non-functional requirements (NFR). Functional requirements define the needs that are directly in contact with stakeholders. In contrast, non-functional requirements describe how the software provides the means to carry out functional requirements[1]. Non-functional requirements are often mixed with functional requirements[2]. Sometimes, there are software requirements specification which is incomplete[3] and ambiguous[4] at the time of writing them. The possibility of inconsistencies in the requirements process makes automatic classification more prone to errors. Then the solution is to find the optimal way to achieve an automated and sound classification. Also, the classification of software requirements is required to perform manual classification requirements, where this process takes a long time, especially for large projects with very complex requirements. SRS is part of the requirement engineering.

Requirements engineering (RE) can be defined as series of activities to explore, evaluate, record, consolidate, revise and adjust the target system, capabilities, quality, limitations, and assumptions. This system must be met according to the questions raised by the following Aspect: RE Importance It is essential to develop effective software and reduce software errors in software development. For example, NFR is ignored during the software development cycle because it is sometimes mixed with FR, or NFR is neglected at a later stage of the software development life cycle. Requirements will be costly and will significantly affect customer satisfaction. Imprecise will be an essential issue in software development. When evaluating alternative architecture and design decisions, it is crucial to identify the NFR as early as possible. To design the system architecture, system architects need NFRs to define constraints, such as security, reliability, performance, scalability, availability, etc. In contrast to the FR, it is difficult for the NFR to handle changes due to a lack of NFR.

In artificial intelligence, text grouping is an ideal and appropriate technique for characterizing content. Because humans have limitations in receiving excessive information, it is impossible to classify documents manually. The simple concept offers several possible things to analyze each document and get an idea of the category. Based on this knowledge, documents that have been processed will be collected according to their respective categories and classes.

Quality is generally defined as the degree to which a series of inherent characteristics meet the requirements[5]. One of the weakest points of the stages of software product quality standards is the lack of guidelines for the formulation of precise quality requirements. The requirements must be clear, complete, consistent, and measurable for an unbiased evaluation of quality. Software quality is how the software can provide and maintain a certain level of service. The level of service required is determined according to the quality model. The software product quality models are provided in ISO / IEC 9126-1 and prepared at ISO / IEC 25010[5]. ISO-9126 is a software quality assurance measurement standard. The basic idea is to define and evaluate software products to determine the internal and external quality and their relationship to quality attributes[6]. ISO-9126 characteristics, namely Function, Reliability, Usability, Efficiency, Maintainability, and Portability. The main advantage of ISO 9126 is that its characteristics apply to every type of software while providing consistent terminology.

There have been a number of works in the effort of identifying the best supervised methods for classifying software requirements[7–12]. Shreda and Hanani[7] compared four classification algorithms using different feature extraction techniques on the NFR of the PURE dataset. The result is the traditional ML method could produce a precision of 87%, and the CNN method outperforms other methods by reaching a precision value of 92%. Li et al.[9] investigating the performance of two decision tree classifiers, namely random forest, and gradient boosting. They tried to find the best classifier for identifying functional requirements and non-functional requirements. Rago et al.[12] selected different direction for classifying textual requirements by considering textual features that represent semantic roles. Rahman et al.[13] identify needs using RNN variants to classify NFR into predetermined categories. The result was that RNN-LSTM outperformed other methods by obtaining a recall value of 71.5%, a precision of 71.7%, and f1-score of 70%, and an accuracy of 71.5%. Fahmi and Siahaan[14] proposed a classification model to detect non-requirement statements on SRS and compared five machine learning methods. The comparison results show that the best model was produced using the SVM method with an average accuracy of 0.96. Tiun et al.[15] perform automatic identification to classify FR and NFR by combining feature extraction and classification methods. Based on the results obtained, the use of FastText seems to be a promising classification model because it received the highest F1 score of 92.8%. Haque et al.[1] proposed an automated NFR classification approach for quality software development by combining machine learning feature extraction and classification techniques. The SGD SVM classifier achieved the best results where precision, yield, F1 score, and accuracy were reported as 0.66, 0.61, 0.61, and 0.76, respectively. Also, the TF-IDF (character level) feature extraction technique described a higher average score than others.

By combining feature extraction, previous researchers used two datasets and different classification methods, such as traditional classification and deep learning. The contribution of our research is to use two datasets, then classify them by comparing the FastText+SVM, FastText+CNN, SVM, and CNN classification methods. This research will be carried out as follows. We start by describing the dataset and justifying its respective parameters. Then we introduce several methods that were compared in this study. Next, we present the measures used to measure classifier performance.

2 | PREVIOUS RESEARCHES

Fahmi and Siahaan[14] proposed a classification model to detect non-requirement statements on SRS and compared five machine learning methods. The machine learning methods compared were Naive Bayes, SVM, random forest, KNN, and decision tree. This comparison aims to determine which method produces the best non-requirement classification model. 14 datasets of software requirements specifications were used to test the five methods. The comparison results show that the best model is produced using the SVM method with an average accuracy of 96%. Rahimi et al.[16] proposed a classification functional requirement (FR) in machine learning using five methods like Naïve Bayes, SVM, Decision Tree, Logistic Regression, and SVC to increase accuracy. In this research which combines several methods, SVM has the best and highest accuracy among other methods. Hakim et al.[2] proposed an evaluated combination of hypernym and synonyms for NFR based on ISO 25010. Lukman used KNN and SVM. Both are based on 'ground truth expert,' and KNN has a precision of about 81.0%, then SVM only 74.6%. Canedo and Mendes[17] proposed a study of text feature extraction with machine learning algorithms classification of software requirements. This experiment uses PROMISE_exp data and logistics regression methods, SVM, Naïve Bayes, and KNN. But this research resulted in poor value due to unbalanced data collection. This classification research had the worse result. So from several papers that we summarize, SVM had high accuracy compared with other traditional machine learning algorithms.

Haque et al.[1] proposed an automated NFR classification approach for quality software development by combining machine learning feature extraction and classification techniques. This process considers the NB, KNN, SVM, and D-tree machine learning algorithms and the comparison of the BoW and TF-IDF feature extraction techniques. The dataset used is the PROMISE dataset. The entire framework process is divided into four steps: Data Pre-processing, Feature Extraction, Train Classifier, and Classifying requirements. The SGD SVM classifier achieved the best results where precision, yield, F1 score, and accuracy were reported as 0.66, 0.61, 0.61, and 0.76, respectively. Also, the TF-IDF (character level) feature extraction technique described a higher average score than others. Tiun et al.[15] perform automatic identification to classify FR and NFR by combining feature extraction and classification methods. Feature extraction used is Doc2Vec, BOW, FastText, and TF IDF. At the same time, the classification methods are LR, SVM, CNN, and NB. The dataset in this study is RE '17 Data Challenge Area 2. Based on the results obtained, FastText seems to be a promising classification model because this model got the highest F1 score of 92.8%. Shreda and Hanani[7] compared four classification algorithms using different feature extraction techniques on the NFR of the PURE dataset. Several feature extractions were used, namely TF, TF-IDF, Word2Vec, and BERT. Machine learning methods used are Naive Bayes, SVM, LR, and CNN. The result is the traditional ML method gets a precision of 87%, and the CNN method outperforms other methods by reaching a precision value of 92%.

Rahman et al.[13] identify needs using RNN variants to classify NFR into predetermined categories, using the PROMISE dataset in their research. Several methods are used to determine the best model, and these methods are RNN-LSTM, RNN-GRU, and CNN. The result was that RNN-LSTM outperformed other methods by obtaining a recall value of 71.5%, a precision of 71.7%, and f1-score of 70%, and an accuracy of 71.5%. Baker et al.[18] classify NFR automatically using two neural network models: ANN and CNN, based on five categories of NFR such as maintainability, operability, performance, security, and usability. They combined two datasets: Requirements Engineering Conference's 2017 Data Challenge and PROMISE. The result indicated that their CNN model effectively classifies NFR and achieved a precision range of 82-94%, recall range of 76-97%, and F-score range of 82-92%. From the two previous studies, the conclusion is that classification using CNN can provide better accuracy.

3 | MATERIAL AND METHOD

This section presents a generic methodology to construct artificial datasets modeling the different characteristics of actual data. In addition, we describe the measurements used to evaluate the quality of the classifiers. The methodology is shown in Figure 1 . In the first stage, we collected two datasets, namely PROMISE and SecReq. PROMISE repository is a collection of datasets provided to the public by the software engineering community to serve researchers and the software industry. Meanwhile, SecReq is a dataset that promotes security requirements, leads to automated research, and increases security awareness[19]. After getting the dataset, we pre-process the raw input data into a suitable format for further analysis. After getting the appropriate data, we perform classification and feature extraction. The results of this classification produce comparisons, and we evaluate them.

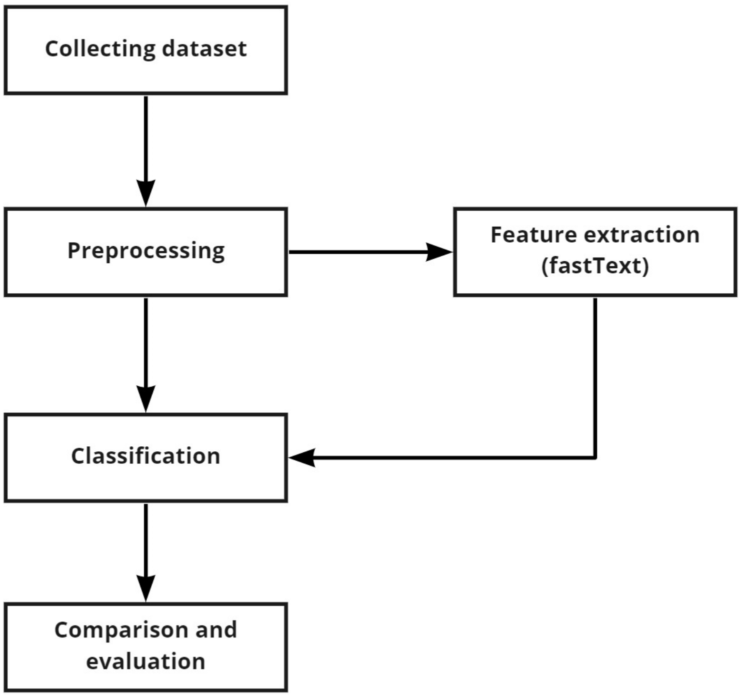


FIGURE 1 The methodology of this study.

TABLE 1 The experimental dataset.

Dataset	Class	Tota Data	Largest class Name	Size	Smallest Class Name	Size
PROMISE	12	625	F	255	PO	1
SecReq (ePurse)	2	124	Security	83	Non-security	41

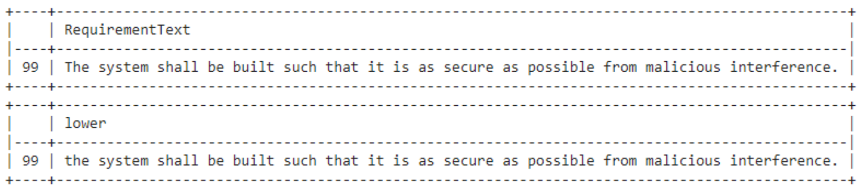
3.1 | Artificial Data

In this study, we used two datasets. The dataset used is PROMISE[13] and SecReq[20]. The PROMISE NFR dataset is commonly used in the community and addressed in the RE'17 Data Challenge[21]. In the dataset PROMISE, the requirement text of the FR category is labeled with 'F' and NFR as 'A, L, LF, MN, O, PE, SC, SE, US, FT, PO.' In our experiment, 625 requirement text is used as the dataset, of which 56% are in NFR class, and the rest is FR class. Secreq has two labels, namely security, and non-security. Houmb et al.[19] collects SecReq datasets to promote security requirements, lead to automated research, and increase security awareness. The SecReq data set consists of three industry SRS documents, namely Common Electronic Purse (ePurse), Customer Premises Network (CPN), and Global Platform Specification (GPS). Each document is broken down into individual requirements, marked as safety-related or non-safety-related. In this study, the dataset used is Secreq ePurse.

Table 1. Dataset

Evaluating the performance of the classifiers In our methodology, system evaluation is achieved by randomly splitting the dataset into train and test subsets. The training set is used for training ML classifiers, while the test set is used only to test the performance of the classifiers. It is worth saying the test dataset has never been used in training (holdout dataset). A general thumb rule that we followed is to use a 70:30 train/test split[7]. Since the results need to be compared with benchmarks, we use accuracy, precision, recall, and F1-score as measurement metrics.

Precision measures the percentage between the number of correctly classified samples to the total sample. Precision is calculated using the correct ratio of the whole classification to the total classification performed.



```
+----+--------------------------------------------------------------------------------+
|    | RequirementText                                                                |
|----+--------------------------------------------------------------------------------|
| 99 | The system shall be built such that it is as secure as possible from malicious interference. |
+----+--------------------------------------------------------------------------------+
+----+--------------------------------------------------------------------------------+
|    | lower                                                                          |
|----+--------------------------------------------------------------------------------|
| 99 | the system shall be built such that it is as secure as possible from malicious interference. |
+----+--------------------------------------------------------------------------------+
```

FIGURE 2 Pre-processing to lowercase.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

A recall is the proportion of positive instances correctly detected by the classifier. It is calculated through the number of times a class is predicted correctly (TP), divided by the number of times the class appears in test data (FN).

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

F-measure is precision and recalls weighted harmonic average. Therefore, the F1 score takes both false positives and false negatives. F1-score can be formulated as:

$$F1Score = 2 \times \frac{Recall \times Precision}{Recall + Precision} \tag{3}$$

It is often easy to combine precision and gain into a single metric called an F1-score (also known as an F-measure) if you need a simple way to compare two classifiers[17]. The F1-score is the average of precision and recall. Whereas the average regular treats all values equally, the harmonic average gives more weight to low values. As a result, classifiers will only get a high F1-score if both recall and precision are high. Accuracy represents a small part of correctly classified observations. It is a more common measure because it calculates the exact number of classifications overall. It is a good measure when the target class in the data is (almost) balanced.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{4}$$

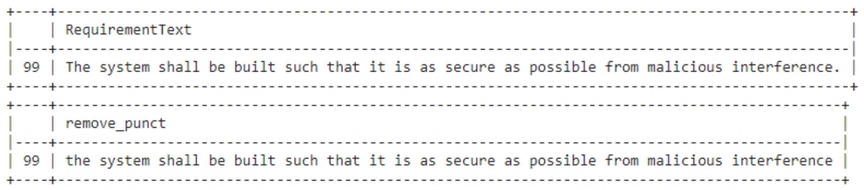
This study compared machine learning algorithms (SVM) and deep learning (CNN) by combining feature extraction techniques (FastText) using two datasets.

4 | RESULTS AND DISCUSSION

This section presents the results and discussion to find out the performance of the compared methods. Several processes include pre-processing, feature extraction, and comparison of classifiers. Pre-processing The first process that is carried out is pre-processing. Pre-processing aims to convert the raw input data into a suitable format for further analysis. The pre-processing used in this research is lowercase, remove punctuation, lemmatization, and stopword removal. Lowercase is the process of converting all letters in a document to lowercase. The following is an example of the application of lowercase, as shown in Figure 2 .

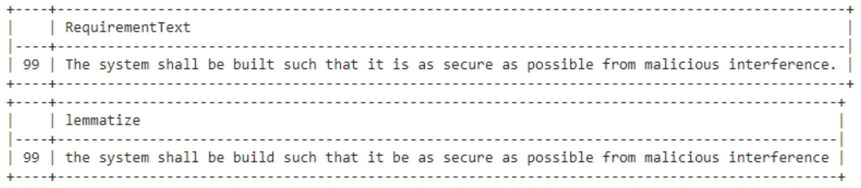
Removing punctuation is removing punctuation marks because punctuation does not influence the classification process. The following is an example of applying removed punctuation, as shown in Figure 3 .

Lemmatization is a technique used to reduce the variety of words in the text of a needs statement by considering the language's vocabulary to apply morphological analysis to the words. The following is an example of the application of lemmatization, as shown in Figure 4 .



```
+----+---------------------------------------------------------------------------------+
|    | RequirementText                                                                 |
|----+---------------------------------------------------------------------------------|
| 99 | The system shall be built such that it is as secure as possible from malicious interference. |
+----+---------------------------------------------------------------------------------+
+----+--------------------------------------------------------------------------------+
|    | remove_punct                                                                   |
|----+--------------------------------------------------------------------------------|
| 99 | the system shall be built such that it is as secure as possible from malicious interference |
+----+--------------------------------------------------------------------------------+
```

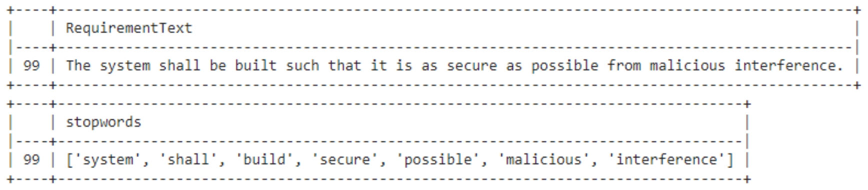
FIGURE 3 Pre-processing to remove punctuation.



```
+----+---------------------------------------------------------------------------------+
|    | RequirementText                                                                 |
|----+---------------------------------------------------------------------------------|
| 99 | The system shall be built such that it is as secure as possible from malicious interference. |
+----+---------------------------------------------------------------------------------+
+----+-------------------------------------------------------------------------------+
|    | lemmatize                                                                     |
|----+-------------------------------------------------------------------------------|
| 99 | the system shall be build such that it be as secure as possible from malicious interference |
+----+-------------------------------------------------------------------------------+
```

FIGURE 4 Pre-processing to reduce the variety of words.

Stopword removal aims to remove common words that usually appear in large numbers and are considered meaningless. The following is an example of the application of stopwords removal, as shown in Figure 5 . After pre-processing, the following process is feature extraction.



```
+----+---------------------------------------------------------------------------------+
|    | RequirementText                                                                 |
|----+---------------------------------------------------------------------------------|
| 99 | The system shall be built such that it is as secure as possible from malicious interference. |
+----+---------------------------------------------------------------------------------+
+----+-------------------------------------------------------------------------+
|    | stopwords                                                               |
|----+-------------------------------------------------------------------------|
| 99 | ['system', 'shall', 'build', 'secure', 'possible', 'malicious', 'interference'] |
+----+-------------------------------------------------------------------------+
```

FIGURE 5 Pre-processing to remove common words.

4.1 | Feature Extraction

Feature extraction is the process of taking features of an object that can describe the characteristics of the object. In this study, the feature extraction used in addition to fasttext is a count vectorizer. CountVectorizer from Scikit-learn is used to calculate word frequency for the process of creating a matrix in which text is converted into a vector by counting the frequency of each word in the document. A word dictionary would be created in this step by counting the number of words in the dataset. Then the words can find with the corresponding frequency. After that, it will produce sentences converted into vectors. Fasttext is a library released by Facebook that can be used for word embedding and has an advantage in syntactic analysis. FastText itself is a development of the Word2Vec library, which has long been known as a library for word embedding. FastText is no longer the most effective and easy but more excellent and efficient[15]. The document/sentence illustration within the FastText version is shaped primarily based totally on the common N-gram functions (FastText is an N-gram primarily based totally vector and now no longer phrase vector as in Word2vec) embedded in a hidden variable. Word2vec is a common method used in text mining and software engineering because word2vec can convert words into vectors. Word2vec will generate a vector with the appropriate adjacent and semantic distances. Two factors will be combined into a vector representation. Figure 6 illustrates how the FastText version represents a sentence (requirement text) with a hidden variable wherein N-gram functions of the sentence, x1..xN are embedded and averaged[15].

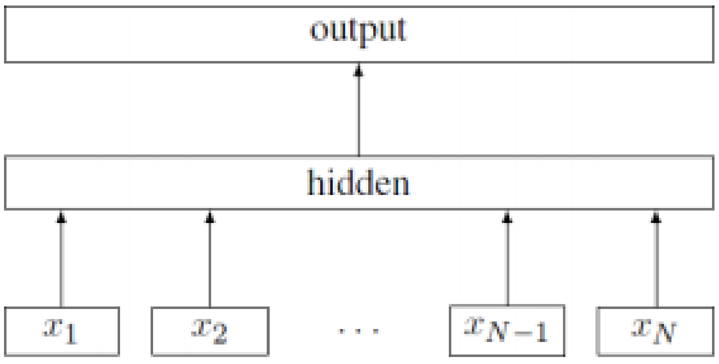


FIGURE 6 The sentences represented by FastText.

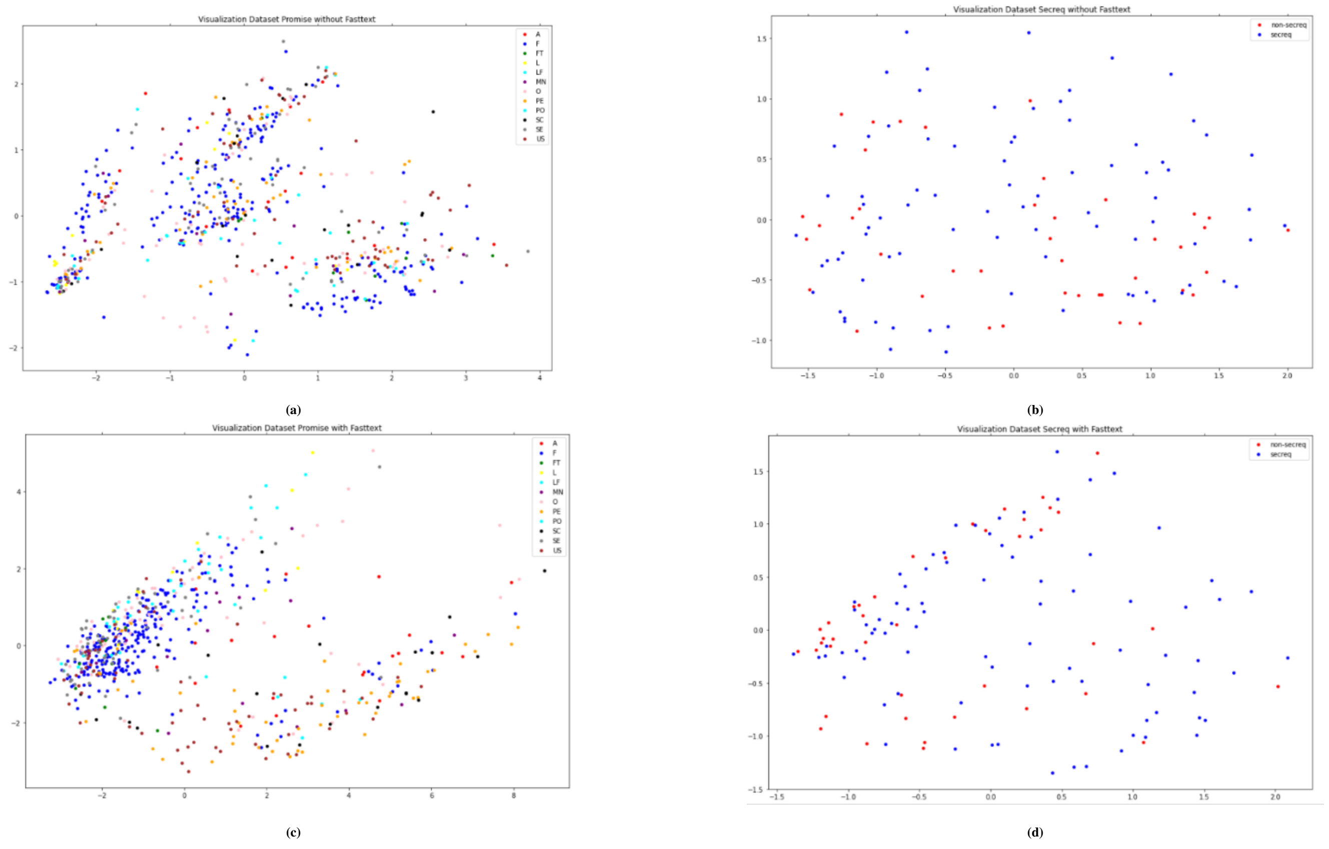


FIGURE 7 The comparison result of the classification methods.

4.2 | Comparison of classifiers

To provide a comprehensive comparison between classifiers, we use the parameters in the algorithm mentioned above. Before entering the method comparison, we will visualize the data using FastText and not FastText. The visualization can be seen in Figure 7 .

Figure 7 shows the datasets that use FastText are more crowded in each class than those that do not use FastText. The SVM method in this study uses linear SVM. For the CNN method, the parameters used are dense layer, relu activation, the epoch

TABLE 2 The result.

Methods	Promise				Secreq			
	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score	Accuracy
SVM	0.98	0.98	0.98	0.98	0.88	0.88	0.88	0.88
SVM Fasttext	0.90	0.90	0.90	0.90	0.90	0.64	0.75	0.76
CNN	0.99	0.99	0.99	0.99	0.94	0.94	0.94	0.94
CNN Fasttext	0.84	0.84	0.83	0.82	0.74	0.72	0.70	0.70

value is 20, and the optimizer uses adam. The number of classes used corresponds to the dataset. Weaknesses that arise in the experiments that have been carried out are in the imbalanced dataset. To overcome these weaknesses, we use duplicating each imbalanced class so that the dataset becomes imbalanced. Table 2 compares SVM, SVM FastText, CNN, CNN FastText on the Promise, and SecReq datasets.

Table 2 explains that the performance of deep learning and machine learning methods without using FastText is better than using FastText. Accuracy on SVM score with FastText on the promise dataset getting a value of 90%. Accuracy on SVM with FastText on the SecReq dataset gets 76%. Accuracy on CNN with FastText on promise dataset scores 82%. Accuracy on CNN with FastText on the SecReq dataset gets 70%. This can be seen from the SVM accuracy score without FastText on the promise dataset, obtaining a value of 98%. Accuracy on SVM without FastText on the SecReq dataset gets 88%. Accuracy on CNN without FastText on promise dataset scores 99%. Accuracy on CNN without FastText on the SecReq dataset gets 94%. CNN method is proven to have a better accuracy rate than SVM. CNN accuracy reaches 99% for the Promise dataset, and CNN accuracy reaches 94% for the SecReq dataset. Our identification of why not using FastText is better than using FastText is because, in FastText, there is a process using a hidden layer. So it is the same as CNN, which uses a hidden layer. Possible parameters of CNN must be tuned. However, the use of FastText on CNN gets a low score because the parameters used are the same as CNN without fasttext. The drawback is the need for fine-tuning the hyperparameters for fasttext CNN.

In this study, there are differences in the accuracy of the promise dataset and the secreq dataset. One possible reason why classification using a promise dataset is better than a secreq dataset is that there are more complex promise datasets than a secreq dataset. When the data becomes more complex, the classification will be more accurate, and the interpretation of features will be better.

5 | CONCLUSION

Machine learning can be applied to detect and classify data. Machine learning has several algorithms, such as Supervised Learning, Unsupervised Learning, and Deep Learning. We should highlight that comparing classifications with multiple methods and datasets is not universal. This means that the classification results with a method on the dataset are not necessarily suitable because each dataset has a different configuration. We use PROMISE and SecReq (ePurse) datasets by applying SVM and CNN methods, and also we combine feature extraction techniques (Fasttext) with two datasets. CNN gets the best score on both datasets. The accuracy obtained by CNN on the PROMISE dataset is 99% and on the Seqreq dataset is 94%.

In future work, researchers may use the optimizer to optimize the hyperparameters of CNN. The goal is to find good hyperparameters when combined with fasttext.

CREDIT

Fajar Baskoro: Conceptualization, Methodology, Formal Analysis, Writing - Review & Editing, and Supervison. **Rasi Aziizah Andrahsmara:** Data Curation, Writing - Original Draft, Validation, and Investigation. **Brian Rizqi Paradisiaca Darnoto:** Data Curation, Writing - Original Draft, Validation, and Investigation. **Yoga Ari Tofan:** Data Curation, Writing - Original Draft, Validation, and Investigation.

References

1. Haque MA, Rahman MA, Siddik MS. Non-Functional Requirements Classification with Feature Extraction and Machine Learning: An Empirical Study. In: 1st International Conference on Advances in Science, Engineering and Robotics Technology 2019 (ICASERT 2019) IEEE; 2019. p. 1–5.
2. Hakim L, Rochimah S, Fatichah C. Evaluasi kombinasi hipernin dan sinonim untuk klasifikasi kebutuhan non-functional berbasis ISO/IEC 25010. Jurnal Teknologi Informasi dan Ilmu Komputer 2019 10;6:491–500.
3. Gu Y, Zhang S, Qiu L, Wang Z, Zhang L. A layered KNN-SVM approach to predict missing values of functional requirements in product customization. Applied Sciences 2021 3;11:2420.
4. Osman MH, Zaharin MF. Ambiguous Software Requirement Specification Detection: An Automated Approach. In: Proceedings - International Conference on Software Engineering IEEE Computer Society; 2018. p. 33–40.
5. Vanicek J. Software quality requirements. Agric Econ 2008;52:177–185.
6. Supriyono S. Penerapan ISO 9126 dalam pengujian kualitas perangkat lunak pada E-book. MATICS 2019 10;11:9.
7. Shreda QA, Hanani AA. Identifying non-functional requirements from unconstrained documents using natural language processing and machine learning approaches. IEEE Access 2021;4:1–22.
8. Solomin AA, Ivanova Bolotova YA. Modern approaches to multiclass intent classification based on pre-trained transformers. Scientific and Technical Journal of Information Technologies, Mechanics and Optics 2020;4(1):532–538.
9. Li LF, Jin-An NC, Kasirun ZM, Piaw CY. An empirical comparison of machine learning algorithms for classification of software requirements. International Journal of Advanced Computer Science and Applications 2019;10(11):258–263.
10. Tóth L, Vidács L. Study of the performance of various classifiers in labeling non-functional requirements. Information Technology and Control 2019;48(3):432–445.
11. Abdel Qader A. A novel intelligent model for classifying and evaluating non-functional security requirements form scenarios. Indonesian Journal of Electrical Engineering and Computer Science 2019 sep;15(3):1578–1585.
12. Rago A, Marcos C, Diaz-Pace JA. Using semantic roles to improve text classification in the requirements domain. Language Resources and Evaluation 2018 sep;52(3):801–837.
13. Rahman MA, Haque MA, Tawhid MNA, Siddik MS. Classifying Non-Functional Requirements Using RNN Variants for Quality Software Development. In: MaLTeSQuE 2019 - Proceedings of the 3rd ACM SIGSOFT International Workshop on Machine Learning Techniques for Software Quality Evaluation, co-located with ESEC/FSE 2019 Association for Computing Machinery, Inc; 2019. p. 25–30.
14. Fahmi AA, Siahaan D. Algorithms comparison for non-requirements classification using the semantic feature of software requirement statements. IPTEK The Journal for Technology and Science 2021 1;31:343.
15. Tiun S, Mokhtar UA, Bakar SH, Saad S. Classification of Functional and Non-Functional Requirement in Software Requirement Using Word2vec and Fast Text. In: Journal of Physics: Conference Series, vol. 1529 Institute of Physics Publishing; 2020. p. 42077.
16. Rahimi N, Eassa F, Elrefaei L. An ensemble machine learning technique for functional requirement classification. Symmetry 2020 10;12:1–26.
17. Canedo ED, Mendes BC. Software requirements classification using machine learning algorithms. Entropy 2020 9;22:1–20.
18. Baker C, Deng L, Chakraborty S, Dehlinger J. Automatic Multi-Class Non-Functional Software Requirements Classification Using Neural Networks. In: Proceedings - International Computer Software and Applications Conference, vol. 2 IEEE Computer Society; 2019. p. 610–615.

19. Houmb SH, Islam S, Knauss E, Jurjens J, Schneider K. Eliciting security requirements and tracing them to design: An integration of common criteria, heuristics, and UMLsec. Requirements Eng 2020;15:63–93.

20. Dekhtyar A, Fong V. RE Data Challenge: Requirements Identification with Word2Vec and TensorFlow. In: Proceedings - 2017 IEEE 25th International Requirements Engineering Conference, RE 2017 Institute of Electrical and Electronics Engineers Inc.; 2017. p. 484–489.

21. Hey T, Keim J, Koziolek A, Tichy WF. NoRBERT: Transfer Learning for Requirements Classification. In: in Proceedings of the IEEE International Conference on Requirements Engineering; 2020. p. 169–179.

How to cite this article: Baskoro F., Andrahsmara R.A., Darnoto B.R.P., Tofan Y.A. (2021), A Systematic Comparison of Software Requirements Classification, *IPTEK The Journal of Technology and Science*, *32(3):184-193*.