International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048
Volume 10, Issue 6, pp 96-102, June-2026
https://doi.org/10.47001/IRJIET/2026.106010

Requirements Quality Evaluation: A Deep Learning Approach

Landry Giraud Wandji T.

Ph.D. Student at the Hochschulinstitut Schaffhausen, Switzerland

***Abstract* - Requirements engineering (RE) is a foundational activity in software development, yet the quality of requirements remains a persistent challenge. Ambiguity, incompleteness, inconsistency, and unverifiability frequently lead to project delays, cost overruns, and system defects. Traditional rules-based evaluation methods, while useful, lack the ability to capture contextual and semantic nuances inherent in natural language requirements. Recent advances in machine learning application for requirements engineering offer new opportunities for automated, scalable, and objective requirements quality assessment. This paper presents a comprehensive deep learning approach for evaluating requirement quality through a defined requirement quality model. Using an appropriate dataset of labeled software requirements, we train and compare several machine learning models, including a Multinomial Naive Bayes (MNB), an Artificial Neural Network (ANN), a Convolutional Neural Network (CNN) and a Recurrent Neural Network (RNN) model. The results demonstrate that the CNN model significantly outperforms the MNB-model as classical machine learning baselines, achieving up to 99% accuracy in multi-label classification. The findings highlight the potential of deep learning to enhance requirements engineering workflows, reduce manual review effort, and improve specification quality.**

Keywords: requirements quality, deep learning, machine learning.

I. INTRODUCTION

Requirements engineering (RE) is widely recognized as one of the most critical phases in the software development lifecycle. High quality requirements directly influence project success, while poorly written requirements are a major source of defects, rework, and stakeholder dissatisfaction [7]. Despite the availability of standards such as IEEE 29148, evaluating requirement quality remains largely manual, subjective, and inconsistent across analysts and organizations.

The increasing complexity of software systems, combined with the growing volume of requirements in large scale projects, has intensified the need for automated quality evaluation. Traditional approaches rely on linguistic heuristics, rule-based systems, or checklists, but these methods do not facilitate to capture semantic meaning, contextual dependencies, and domain specific nuances.

Recent breakthroughs in natural language processing (NLP), particularly transformer based deep learning models such as BERT [3], have revolutionized text understanding. These models excel at capturing contextual semantics and have achieved state of the art performance across numerous NLP tasks. Their potential for requirements engineering is significant but still underexplored.

This paper proposes a deep learning approach for automated requirements quality evaluation. We develop a multi-label classification framework that assesses requirements across four key quality dimensions: clarity, completeness, consistency, and verifiability. The contributions of this work include:

- A deep learning framework for multi-dimensional requirements quality evaluation.
- An appropriate dataset of annotated requirements aligned with IEEE 29148 quality criteria.
- A comparative evaluation of deep learning models against classical machine learning baselines.
- An analysis of model performance, and practical implications for RE workflows.

II. RELATED WORK

Early research in requirements quality assessment focused on rule based and checklist driven methods. Tools such as QuARS [4] has been developed to support a methodology for the analysis of natural language requirements based on a quality model, addressing a relevant part of the interpretation problems that can be approached at linguistic level. Another tool called ARM?? detect ambiguity and linguistic defects using handcrafted rules. While effective for specific patterns, these systems require extensive maintenance and lack generalization across domains.

Classical machine learning methods, including SVMs and Naive Bayes, have been applied to requirement classification and defect detection [9]. They present the design of a tracing tool for automatically recovering traces between JIRA issues

International Research Journal of Innovations in Engineering and Technology (IRJIET)
ISSN (online): 2581-3048
Volume 10, Issue 6, pp 96-102, June-2026
https://doi.org/10.47001/IRJIET/2026.106010

and commits in a model-driven development (MDD) context. Using process and text-based data, they created 154 features to train a ML classifier. This classifier was then validated using four real MDD industry datasets. They were able to get an average F2-sore of 73.48 with the best tested configuration. An F0.5-score of 77.32 was obtained in the scenario of automatically maintaining traces of a current project. However, these models rely heavily on feature engineering and cannot capture long range dependencies or contextual semantics.

Recent studies have explored deep learning for RE tasks such as ambiguity detection [8] and also incorporates data augmentation using back translation to mitigate the class imbalance by generating diverse samples of the minority class and improving the model's generalizability. In addition to the proposed GloVeBiLSTM model, they conducted an ablation study to evaluate the impact of different components by comparing it with alternative configurations, including GloVe-CNN, GloVe-CNN-RNN-LSTM, and GloVe-LSTM. This study provided insights into the contributions of each component, with GloVe-BiLSTM outperforming other models, achieving over 90% accuracy and superior performance across precision, recall, and F1-score metrics. Specifically, GloVe-BiLSTM attained a precision of 92.04%, recall of 91.52%, F1-score of 91.65%, and an accuracy of 92.06%.

Another approach to detect ambiguity in software requirements has been presented by [1]. They have used meta-heuristic algorithms to choose the best value of hyperparameters of their deep learning algorithm. The publicly available Fault-Prone SRS dataset was utilized to train the models. This dataset was also used to evaluate the performance of the proposed algorithm in terms of F1score, accuracy, and other statistical metrics. As results, the BERT-BiLSTM model outperformed other models in classifying and detecting ambiguities in requirement specification documents, achieving an F1-score and higher than 81% accuracy. Transformer models show strong potential but are often limited to single quality dimensions or small datasets. Our work extends this by providing a holistic, multi-label evaluation framework.

III. METHODOLOGY

3.1 Dataset

We constructed a dataset of 5,000 functional and non-functional requirements sourced from industrial specifications, academic repositories, and synthetic generation. Using the proposed quality evaluation metric proposed by [6], each requirement has been labeled as presented in the figure 1. The target will be the assessment of the relevant elements in each sentence.

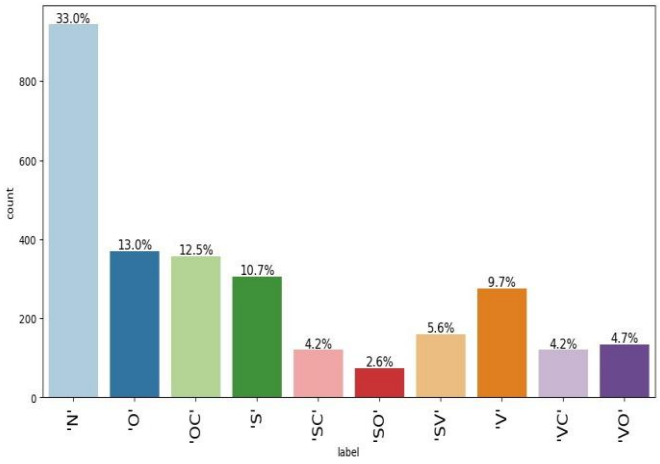

Figure 1: Overview of the labeled requirements

The explanation of the requirements label is the following:

- "N": (Normal) for well written requirement, without missing of a relevant element
- "S": (Subject) for a missing subject in the requirements sentence
- "V": (Verb) for a missing verb in the requirements sentence
- "O": (Object) for a missing object in the requirements sentence
- "SV": (Subject Verb) for a missing subject and verb in the requirements sentences
- "SC": (Subject Complement) for a missing subject and complement in the requirements sentence
- "SO": (Subject Object) for a missing subject and verb in the requirements sentence
- "VO": (Verb Object) for a missing verb and object in the requirements sentence
- "VC": (Verb Complement) for a missing verb and complement in the requirements sentence
- "OC": (Object Complement) for a missing object and complement in the requirements sentence

3.2 Requirements Quality evaluation

A quality evaluation model has been build based on the description presented by [6]. Each element of the sentence is count with a value 1 for present element and 0 for missing element if applicable. The applicable elements are actor (subject), action (verb), object, destination of the action (complement). The non applicable element is a complement to refine the action. The figure 2 show an example of sentences evaluated using this model.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 6, pp 96-102, June-2026

https://doi.org/10.47001/IRJIET/2026.106010

	Element	Applicability	Text example 1	(score)	Text example 2	(score)	Text example 3	(score)
1	Actor	1	The system	1		0	The system	1
2	Action	1	Shall display	1	Shall encrypt	1		0
3	Object of the action	1	Error message	1	All outbound transaction	1	The configuration	1
4	Refine of action	0		0		0		0
5	Destination of /constraint to the action	1	In case of invalid input	1		0	Before it is	1
RQ			IRQ1	0,8	IRQ2	0,4	IRQ3	0,6

Figure 2: Example of requirements sentence evaluation

Based on the elements presents in the requirement sentence, a score for each requirement is build called Individual Requirement quality (IRQ). The sum of all the individual requirement quality is divided by the number of all the requirements to build a metric called Requirement Quality (RQ). The following equation (1) shows le calculation of the requirement quality.

$$RQ = \frac{\sum_{i=1}^{n} IRQi}{n} \qquad (1)$$

The results will be evaluated based on the definition mentioned by [6] as showed in the figure 3.

Metric	Very poor set of requirements, requiring substantial development	Fair set of requirements, may just be suitable for purposes of solicitation, depending on the SOW and type of contract envisaged	Requirements at SRR suitable for carrying forward into development	Requirements suitable for Critical Development
RQ-	0.01-0.3	0.3-0.7	0.85-0.99	0.99+

Figure 3: Requirement quality evaluation

3.3 Machine learning classifier

The next step of this work will be to build and train machine learning classifier for the quality evaluation of the requirements sentences. A classical machine learning model and deep learning models will be built. For this study a similar approach as presented in [5] will be conducted. Multinomial Naive Bayes (MNB) will be investigated and his performance will be compared to those of deep learning techniques. Artificial Neural Network (ANN), Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) will be implemented and tested in the first stage. In the second stage the best of all the models will be test on four requirements chosen randomly in the dataset.

The models performance will be evaluated based on metrics as accuracy, precision, F1-score and recall. This performance metric shows the aptitude of the machine learning models to classify each requirement in the corresponding category. Depending on the quality of the data used to train the models, the performance of the trained model will be replicable on different datasets. The best of all the trained models will be saved to be incorporated in the architecture of a requirements engineering tool.

IV. RESULTS

4.1 Results of the MNB model

The classical machine learning model used as baseline for this approach is the Multinomial Naives Bayes (MNB) model. The model has been trained and tested using the data sets presented in 3.1. The figure 4 shows the result the classification of the MNB model.

	precision	recall	f1-score
'N'	0.8182	0.8095	0.8138
'O'	0.9459	0.9459	0.9459
'OC'	0.8642	0.9859	0.9211
'S'	0.5000	0.4918	0.4959
'SC'	1.0000	0.9167	0.9565
'SO'	0.7895	1.0000	0.8824
'SV'	0.9630	0.8125	0.8814
'V'	0.9138	0.9636	0.9381
'VC'	0.7692	0.8333	0.8000
'VO'	0.9444	0.6296	0.7556
accuracy			0.8322
macro avg	0.8508	0.8389	0.8391
weighted avg	0.8346	0.8322	0.8305

Figure 4: Overall results of the MNB model

The results show relatively strong precision and recall for several classes, indicating that the model captures some linguistic patterns effectively. However, noticeable variability across categories highlights the model's sensitivity to class imbalance and overlapping feature distributions. The overall accuracy of 83% suggests that MNB provides a solid baseline but lacks robustness for more nuanced requirement types. These findings confirm that while MNB is computationally efficient, but more advanced models are needed for consistent, high quality requirements evaluation. This result is also highlighted by the confusion matrix showed in the figure 5.

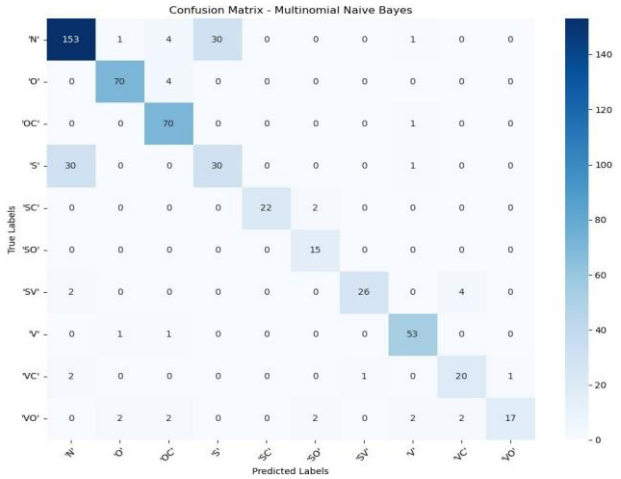

Figure 5: Conclusion matrix results of the MNB model

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 6, pp 96-102, June-2026

https://doi.org/10.47001/IRJIET/2026.106010

4.2 Results of the ANN model

An Artificial Neural Network (ANN) model has been also evaluated in this approach on the same data as the MNB model. After training and testing of the model, his performance has been observed in the training and in the validation phase as well. The evolution of training and validation accuracy has been reported across epochs, showing steady convergence as presented in the figure 6.

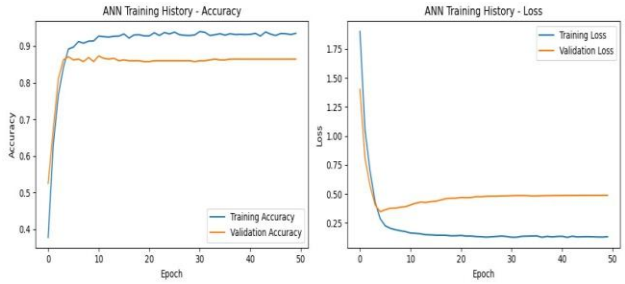

Figure 6: Overall results of the ANN model

Training accuracy rises rapidly and stabilizes above 90%, indicating effective learning of the model. Validation accuracy plateaus around 85%, suggesting good generalization with minor performance gaps. The loss curves show a sharp initial decrease, with training loss dropping below 25% as the model optimizes. Validation loss stabilizes at a higher level than training loss, hinting at mild overfitting in later epochs. Overall, the training history demonstrates a well-behaved learning process, with consistent improvement and stable convergence patterns. This result is also illustrated by the confusion matrix showed in the figure 7.

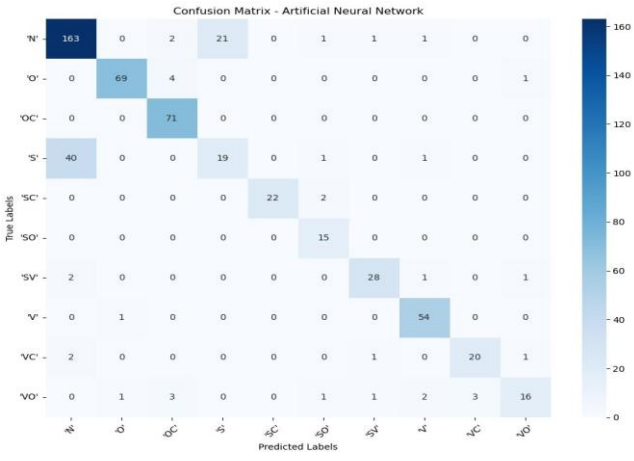

Figure 7: Confusion matrix results of the ANN model

4.3 Results of the CNN model

A Convolutional Neural Network (CNN) has been also evaluated in this approach on the same data as the MNB model. After training and testing of the model, his performance has been observed in the training and in the validation phase as well. The evolution of training and validation accuracy has been reported across epochs, showing steady convergence as presented in the figure 8.

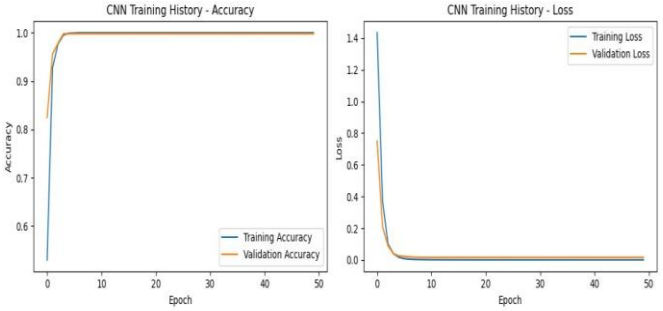

Figure 8: Overall results of the CNN model

The training accuracy reaches close to 1.0, indicating that the model learns the underlying patterns in the dataset very effectively. Validation accuracy follows a similar trend, suggesting strong generalization with minimal performance divergence. The loss curves show a steep decline during early epochs, reflecting efficient optimization and stable gradient behavior. Validation loss remains closely aligned with training loss, indicating limited overfitting throughout the training process. Overall, the training history demonstrates a well-behaved learning trajectory, with consistent improvements and robust convergence across epochs. The confusion matrix showed in the figure 9 also highlighted this result.

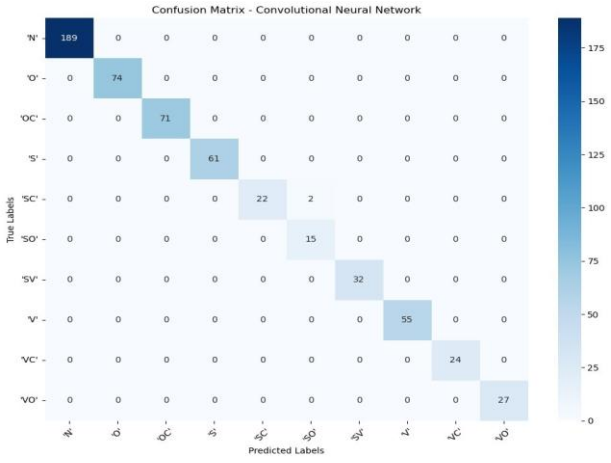

Figure 9: Overall results of the CNN model

4.4 Results of the RNN model

A Recurrent Neural Network (RNN) has been also considered in this approach on the same data as the MNB model. After training and testing of the model, his performance has been observed in the training and in the validation phase as well. The evolution of training and validation accuracy has been reported across epochs, showing steady convergence as presented in the figure 10.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 10, Issue 6, pp 96-102, June-2026

https://doi.org/10.47001/IRJIET/2026.106010

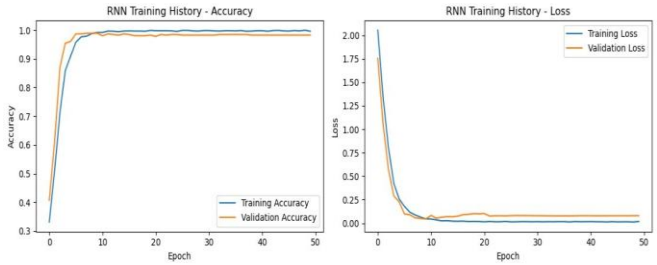

Figure 10: Overall results of the RNN model

Accuracy stabilizes near 98% for both curves, indicating that the model quickly learns the underlying patterns in the dataset. The corresponding loss curves show a steep decline, reflecting efficient optimization and rapid convergence. Training and validation loss remain closely aligned, suggesting minimal overfitting throughout the training process. The smooth plateauing of both metrics demonstrates stable learning behavior after the early rapid improvement phase. Overall, the training history confirms that the RNN achieves strong performance with consistent generalization across epochs. The confusion matrix showed in the figure 11 also describes this result.

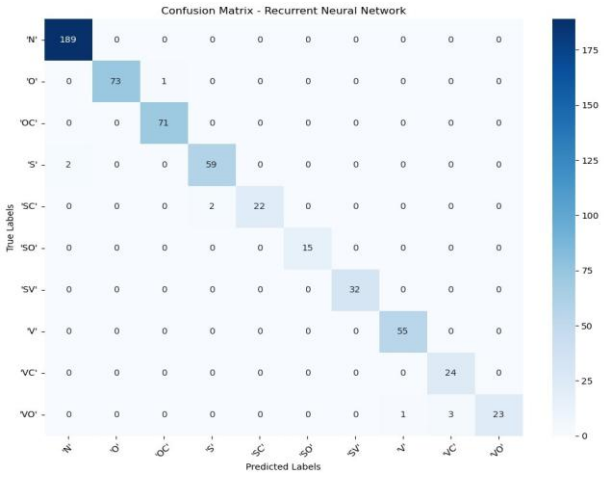

Figure 11: Overall results of the RNN model

4.5 Summary of the results

A comparative analysis of the four machine learning models CNN, RNN, ANN, and MNB across multiple evaluation metrics is presented by the figure 12. The accuracy comparison shows a clear performance hierarchy, with CNN achieving the highest score (99.7%), closely followed by RNN (98.4%), while ANN and MNB trail significantly at approximately 83.4% and 83.2%. The macro precision plot reinforces this trend, as both CNN and RNN exhibit near perfect precision, indicating highly reliable positive predictions across all classes. In contrast, ANN and MNB display noticeably lower precision values, reflecting greater variability in their classification behavior.

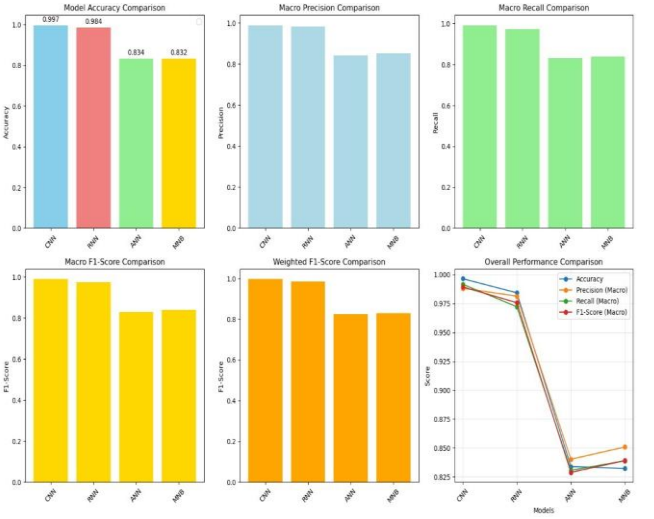

Figure 12: Summary of the models comparison

Similarly, the macro recall comparison demonstrates that CNN and RNN capture the majority of relevant instances with exceptional consistency, whereas ANN and MNB show reduced recall, suggesting more frequent missed detections. The macro F1-score graph integrates these effects, again highlighting the superiority of CNN and RNN, whose balanced precision recall performance results in F1-scores close to 1.0. ANN and MNB maintain moderate F1-scores around 85%, confirming their limitations in handling complex linguistic patterns. The following table summarizes the performance of each model.

Table 1: Performance Comparison of CNN, RNN, ANN, and MNB Models

Model	Accuracy	Precision (Macro)	Recall (Macro)	F1-Score (Macro)
CNN	0.99	0.98	0.99	0.98
RNN	0.98	0.98	0.97	0.97
ANN	0.83	0.84	0.83	0.82
MNB	0.83	0.85	0.83	0.83

The weighted F1-score comparison further emphasizes the robustness of CNN and RNN, as their performance remains stable even when accounting for class imbalance. Finally, the overall performance chart synthesizes all metrics, clearly illustrating that CNN and RNN consistently outperform ANN and MNB across every evaluation dimension. Collectively, the figure demonstrates that deep learning architectures particularly CNNs and RNNs provide substantial advantages for requirements quality classification, offering superior accuracy, generalization, and reliability compared to traditional or shallow models. The prediction example using the CNN as best of all the models shows his ability to well classify each requirement in the corresponding class as show in the figure 13.

ISSN (online): 2581-3048
Volume 10, Issue 6, pp 96-102, June-2026
https://doi.org/10.47001/IRJIET/2026.106010

```
Requirement: the module the firmware descriptor...
Actual: 'V'
Predicted: 'V' (Confidence: 1.0000)
Top 3 predictions:
  - 'V': 1.0000
  - 'VO': 0.9436
  - 'VC': 0.9428
--------------------------------------------------

Requirement: the system shall allow editing of existing records...
Actual: 'N'
Predicted: 'N' (Confidence: 1.0000)
Top 3 predictions:
  - 'N': 1.0000
  - 'O': 0.7921
  - 'OC': 0.2994
--------------------------------------------------

Requirement: the system shall sanitize inbound hl7 messages before ingestion...
Actual: 'N'
Predicted: 'N' (Confidence: 1.0000)
Top 3 predictions:
  - 'N': 1.0000
  - 'O': 0.8223
  - 'VO': 0.0367
--------------------------------------------------

Requirement: the banking api must send to...
Actual: 'OC'
Predicted: 'OC' (Confidence: 1.0000)
Top 3 predictions:
  - 'OC': 1.0000
  - 'S': 0.9782
  - 'N': 0.9728
--------------------------------------------------

Requirement: the radiology console during modality specific routing...
Actual: 'VO'
Predicted: 'VO' (Confidence: 0.9997)
Top 3 predictions:
  - 'VO': 0.9997
  - 'V': 0.6597
  - 'SV': 0.5345
--------------------------------------------------
```

Figure 13: Summary of the models comparison

V. Conclusion

The literature review on requirements quality analysis shows a gap to be field regarding the application of machine learning methods on quality assessment of software requirements. The most research work have been conducted on the requirements traceability using machine learning methods, but the requirements quality evaluation is actually done rather manually using the existing tools available in the requirement engineering community. Some approaches using natural language processing have been investigated on identification of ambiguity and linguistic defects in requirements, but the results are showing that the work to be done on that area still remain important.

The approach proposed by this work to evaluate software requirements quality shows a structured methodology based on machine learning methods trained to recognized missing elements into a requirement sentence. This conduct to build a requirements evaluation metric, which will be extend on the whole requirements datasets to give an evaluation of the customer requirements. The results demonstrate that the Convolutional Neural Network models significantly outperform classical baseline. The using of stratify sampling further improves performance of the models, highlighting the impact of class imbalance on the results of machine learning models.

This research demonstrates the feasibility and effectiveness of deep learning for automated requirements quality evaluation. The Convolutional Neural Network models provide consistent, scalable, and objective assessments across multiple quality dimensions; however, Challenges still remain in other topics like the optimization of the models performance using a larger amount of training data and the evaluation of other quality aspects of a requirement like consistency, connectivity, testability, feasibility, traceability, which were not in the scope of this work.

Future work includes expanding the datasets to make it consistent to train machine learning models and make the datasets available for the software requirements engineering community. Another investigation to be conducted will be to train and test other advanced techniques like BERT-model on these datasets. The application of machine leaning models to evaluate other aspects of a requirement like testability, feasibility can also be included in the next step of this work. The integration of the CNN-model as best performer in a tool for quality evaluation of software requirements can also be a first step for a prototype that can be deployed in industrial RE environments.

REFERENCES

[1] Younes Abdeahad *et al.* “Optimizing Deep Learning Models for Detecting Ambiguities in Software Requirements: Harnessing BERT, Random Search, and Bi-LSTM”. *In: Computational Intelligence* 41.6 (2025), e70156.

[2] Nathan Carlson and Phil Laplante. “The NASA automated requirements measurement tool: a reconstruction”. *In: Innovations in Systems and Software Engineering* 10.2 (2014), pp. 77–91.

[3] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. *In: Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers).* 2019, pp. 4171–4186.

[4] Fabrizio Fabbrini et al. “The linguistic approach to the natural language requirements quality: benefit of the use of an automatic tool”. *In: proceedings 26th annual NASA Goddard software engineering workshop. IEEE.* 2001, pp. 97–105.

[5] Wandji T. Landry G. “A Multiclass Classification of Software Security Requirements: A Deep Learning Approach”. *In: IRJIET* 10 (2026).

[6] Robert J. Halligan. “Requirements Quality Metrics: The Basis of Informed Requirements Engineering Management”. *In: CSESAW ’93* (2017).

ISSN (online): 2581-3048
Volume 10, Issue 6, pp 96-102, June-2026
https://doi.org/10.47001/IRJIET/2026.106010

[7] Hubert F Hofmann and Franz Lehner. "Requirements engineering as a success factor in software projects". *In: IEEE software* 18.4 (2001), p. 58.

[8] Rahat Izhar, Shahid N Bhatti, and Sultan A Alharthi. "Bridging precision and complexity: A novel machine learning approach for ambiguity detection in software requirements". *In: IEEE Access* (2025).

[9] RS Rasiman. "A machine learning approach for requirements traceability in model-driven development". *MA thesis.* 2021.

Citation of this Article:

Landry Giraud Wandji T. (2026). Requirements Quality Evaluation: A Deep Learning Approach. *International Research Journal of Innovations in Engineering and Technology - IRJIET*, 10(6), 96-102. Article DOI https://doi.org/10.47001/IRJIET/2026.106010

****

