IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Modal Verb Based Software Requirements Prioritization Through Classification - A Novel Approach Using Machine Learning And Natural Language Processing

[1]Suchetha Vijayakumar, [2]Suresha D

[1]Research Scholar, Srinivas University and Associate Professor, St Aloysius (Deemed to be University), Mangalore, India

[2]Professor, Srinivas Institute of Technology, Mangalore, India

Abstract: Effective and systematic prioritization of functional requirements is critical and crucial for successful software development, especially in complex projects. Most traditional methods often overlook the linguistic nuances of requirement specifications, such as modal verbs, which convey varying levels of priority. This study explores the integration of modal verb analysis with machine learning classifiers to automate the prioritization process. Functional requirements are classified into High, Medium, and Low priority levels based on the presence of modal verbs such as must, will, should, would, may, can, might. Using a dataset of 375 requirement descriptions from 25 different projects, we employed TF-IDF vectorization for feature extraction and evaluated various classifiers including Random Forest, Logistic Regression, Naive Bayes, and Linear SVM. Linear SVM emerged as the best-performing model, achieving 96% accuracy with superior precision, recall, and F1-score metrics. This work demonstrates the potential of Natural Language Processing features and Machine Learning to enhance requirement prioritization, making software development workflows more efficient and automated.

Index Terms - Functional Requirement Prioritization, Modal Verbs, Machine Learning, Natural Language Processing, Requirement Engineering, TF-IDF Vectorization, Linear SVM, Classifier Comparison, Software Development, Automated Prioritization.

1. Introduction

Prioritizing functional requirements is an important milestone of successful software development, that enables teams to allocate resources efficiently, meet stakeholder expectations, and deliver high-quality products on time [1]. Traditionally, in complex projects, the huge volume of requirements can make manual prioritization tedious, error-prone, and subjective. Automating the prioritization process often offers significant advantages, ensuring consistency, scalability, and precision in identifying high-priority needs. [2]

As an additional aspect of MOSCoW method of requirement prioritization [3], we can make use of the modal verbs, such as must, will, should, would, may, can, might which apparently indicate varying levels of priority. These modal verbs provide linguistic cues that can be leveraged to classify requirements into categories like High, Medium, and Low priority. The existing traditional prioritization methods fail to work on these nuances, leading to inconsistent prioritization outcomes.

This study is carried out to find the most effective machine learning model for automating modal verb-based requirement prioritization. By combining natural language processing techniques, such as TF-IDF vectorization [4], with various machine learning classifiers, this work aims to evaluate their performance in accurately classifying requirements as High, Medium or Low. Specifically, this research seeks to determine

the best-performing model among Random Forest, Logistic Regression, Naive Bayes, and Linear SVM, offering insights into their applicability and effectiveness in the context of software requirement prioritization.

This Research paper is organized as follows. The section that follows contains findings of few papers that are related to the topic and work. The third section contains the methodology of the work. This also includes the dataset description, steps involved, classifiers considered and evaluation metrics. Next section of the research paper consists of the libraries and modules used for coding. Results and Discussion follows. The paper concludes with a conclusion and also contains a mention about the future work associated.

2. Research Objectives

This study aims to:

i. Propose an innovative approach for requirement prioritization
ii. Evaluate the effectiveness of machine learning classifiers
iii. Demonstrate the potential of linguistic features in prioritization
iv. Provide insights into efficient requirement management

3. Related work

In their paper, Hujainah, F. etal [5] introduce “SRPTackle”, a semi-automated technique for prioritizing requirements in software system projects. It addresses the challenges of scalability and subjectivity in requirements prioritization, especially in large-scale projects. The authors validated “SRPTackle” through case studies and experiments with software projects. Results demonstrated improved efficiency, reduced prioritization time, and consistency in prioritization outcomes. The semi-automated approach was especially beneficial in managing large sets of requirements. Talele, P. etal [6], focuses on the application of machine learning techniques to classify and prioritize software requirements efficiently. It aims to address the challenges of manual prioritization and classification in large-scale software development. Machine learning models can effectively handle the classification and prioritization of software requirements, even in large datasets. The use of NLP enhances the system's ability to understand and process requirements written in natural language. Automating these processes reduces manual effort and ensures consistency in decision-making. According to Rahamathunnisa, U etal [7] , artificial intelligence (AI) techniques can address risks associated with software requirements during the software development lifecycle. The paper focuses on risk identification, analysis, and mitigation to enhance project success rates. This paper highlights the effectiveness of AI in addressing software requirement risks, providing actionable solutions to improve requirement quality and reduce project failures. It advocates for the integration of AI with emerging technologies to enhance software development practices in Industry 4.0. Kadhim, A. I. [8] in his paper provides a comprehensive survey of supervised machine learning techniques used for automatic text classification. The authort examines existing methods, their applications, and challenges in processing and categorizing textual data. The paper serves as a foundational resource for understanding supervised machine learning methods in text classification. It underscores the importance of algorithm selection, feature engineering, and evaluation metrics in achieving effective and accurate classification.

4. Methodology

The methodology employed in this research aims to prioritize functional requirements in software requirement specifications using machine learning techniques, leveraging modal verbs as a key feature. To achieve this, we adopted a structured approach that involves dataset preparation, feature engineering, model training, and performance evaluation.

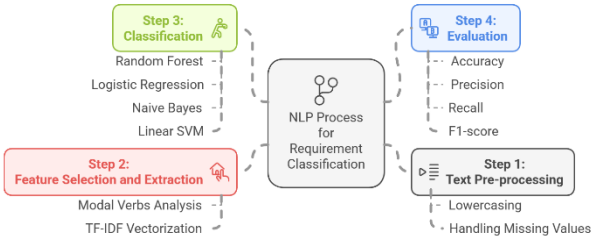

Figure. 1 Workflow diagram for Requirement Classification and Prioritization

3.1 Dataset Description

We have used a dataset that contains software requirements along with their project names, requirement, assigned priorities, and actual ordering. This structure provides a comprehensive base for testing the efficacy of prioritization techniques. It contains Software Requirements of 25 different projects and contains 375 rows of data in the form of requirements. All these are functional requirements of different projects done by the Post Graduate students. Here is a brief explanation of the columns:

Table 1. Brief Description of the dataset

Column name	Description	Purpose
Project Name	Represents the name of the project to which the requirement belongs.	Allows project-specific prioritization and comparison.
Requirement ID	A unique identifier (e.g., R1, R2, ...) for each requirement within a project.	Ensures traceability and differentiation of individual requirements.
Requirement	A textual description of the functional requirement.	Often includes modal verbs (e.g., "must," "should," "may") that help infer priority levels.
Priority	Indicates the assigned priority level (e.g., High, Medium, Low) of the requirement.	Derived based on modal verbs that are manually annotated.
Actual Order	Specifies the rank or order of the requirements after prioritization.	Useful for validating the model's predictions against the true order

4.1 Text Pre-processing

To prepare the requirements for analysis, a series of pre-processing steps were applied, including converting Requirements to lowercase to ensure uniformity. A routine was initiated to check if the data contained any missing values and they were handled suitably.

4.2 Feature Engineering

In software requirements engineering, accurately determining the priority of requirements is a critical task. Traditional requirement prioritization methods rely on subjective assessments, which may lead to inconsistencies. To automate and enhance this process, we leverage Natural Language Processing (NLP) techniques, specifically feature extraction from textual requirements.

4.2.1 Modal Verb Extraction

Modal verbs such as must, will, should, would, may, can, might are integral to this research as they signal varying levels of priority in requirements [9] as they serve as linguistic indicators of obligation, possibility,

and necessity. These modal verbs play a crucial role in determining the priority of requirements, as different levels of obligation correlate with varying levels of requirement importance. Table 2 presents the categorization of modal verbs into priority levels.

To extract this information from the dataset, each requirement was analysed for the presence of modal verbs. The following algorithm was applied:

Step 1: Convert the requirement text to lowercase to ensure case insensitivity.
Step 2: Check for the presence of modal verbs using pattern matching.
Step 3: Assign a priority label based on the strongest modal verb present.
Step 4: If multiple modal verbs exist, the most significant one determines the priority.
Step 5: If no modal verb is found, a default priority of Low is assigned.

Table 2: Modal verbs, their meaning and Requirement Category mapping

Modal Verb	**Intuition/Meaning**	**Requirement Category**
Must	Necessity/obligation	High
Will	Certainty/intention	
Would	Conditional certainty	
Should	Probability/advise	
May	Possibility/permission/possible	Medium
Can	Possibility/permission	
Might	Low possibility/polite permission	Low
Could	Possibility/permission/possible	

4.2.2 TF-IDF Vectorization

In natural language processing (NLP), text-based features need to be transformed into numerical representations for machine learning models to process. TF-IDF (Term Frequency-Inverse Document Frequency) is a widely used text vectorization technique that converts textual data into numerical values by capturing the importance of words in a given document relative to the entire dataset.[10]

In software requirement prioritization, textual descriptions of requirements contain varying degrees of importance based on the presence of modal verbs (e.g., "must", "should", "may"). A simple Bag of Words (BoW) model only considers the presence or absence of words, ignoring their importance across documents. TF-IDF improves upon this by:

- Reducing the weight of common words that appear frequently but do not contribute to meaning (e.g., "the", "is").
- Increasing the importance of rare but significant words (e.g., "must", "should", "critical").
- Enhancing the distinction between high, medium, and low-priority requirements by emphasizing meaningful terms.

The TF-IDF score of a term t in a document d is calculated as:

$$TF - IDF(t,d) = TF(t,d) \times IDF(t)$$

Where:

Term Frequency (TF) measures how frequently a term appears in a document

$$TF(t,d) = \frac{Number\ of\ times\ term\ t\ appears\ in\ the\ document\ d}{Total\ number\ of\ terms\ in\ document\ d}$$

Inverse Document Frequency (IDF) reduces the weight of common words and increases the weight of rare words

$$IDF(t) = \log\left(\frac{N}{DF(t)} + 1\right)$$

where N is the total number of documents, and DF(t) is the number of documents containing term t

4.3 Classification Models

To classify software requirements into High, Medium, and Low priority levels, we evaluated four widely used machine learning classification algorithms: Random Forest, Logistic Regression, Naïve Bayes, and Linear Support Vector Machines (SVM). These models were selected due to their proven effectiveness in text classification tasks and their ability to handle diverse dataset characteristics efficiently[11]. The reasons behind choosing the above models are tabulated in Table 3.

Table 3: Comparison of Characteristics of Classification Algorithms used

Classification Algorithm	Characteristic feature
Random Forest	A method that uses multiple decision trees to make predictions. It also gives high accuracy and control overfitting [12].
Linear Regression	A linear model widely used for binary and multiclass classification problems, offering interpretability and robustness [13].
Naive Bayes	A probabilistic model that assumes independence between features, making it particularly efficient for text classification [14].
Linear Support Vector Machine	A robust linear classifier that separates data points with a hyperplane, optimized for text-based tasks due to its ability to handle high-dimensional data [15].

Each classifier was trained and tested on the preprocessed dataset, where the requirements were vectorized using TF-IDF before being fed into the models. The models were implemented using Scikit-Learn, a widely used machine learning library in Python. The dataset was split into training (80%) and testing (20%) subsets to evaluate model performance.

4.4 Evaluation Metrics

The classifiers were assessed using standard performance metrics, including accuracy, precision, recall, and F1-score [16]. These metrics provide a holistic understanding of each model's strengths and limitations in predicting requirement priorities.

5. Experimental setup:

This section outlines the experimental environment, tools used, and the configuration of the machine learning models employed in the study. The experiments were conducted using Python as the programming language in the Google Colab environment, which provides a cloud-based platform with access to GPUs for accelerated computation. The following tools and libraries were utilized:

- Pandas: For data manipulation and preprocessing.
- Scikit-learn: For feature extraction (TF-IDF), machine learning model implementation, and performance evaluation metrics.
- Joblib: For saving and loading the trained models and vectorizer.
- Google Colab Drive Integration: For data storage and model persistence.

The dataset was split into training and testing subsets to ensure a robust evaluation of the models. The split ratio was set at 80:20, meaning 80% of the data was used for training the models, and 20% was reserved for testing their performance. A fixed random seed (random_state=42) was used to ensure reproducibility of the results.

6. Results and Discussion

The results of the classifier evaluation for prioritizing functional requirements using modal verbs are presented in the Table 4. The Qualtitative Analysis summary of the same is given in Table 5. Four classifiers—Random Forest, Logistic Regression, Naive Bayes, and Linear SVM—were assessed using accuracy, precision, recall, and F1-score as evaluation metrics. These metrics were calculated to provide a comprehensive view of each model's performance in predicting requirement priorities.

Table 4: Quantitative Evaluation Metrics

Classifier	Accuracy	Precision	Recall	F1-Score
Random Forest	0.92	0.937	0.837	0.879
Logistic Regression	0.76	0.85	0.497	0.536
Naive Bayes	0.68	0.227	0.333	0.27
Linear SVM	**0.96**	**0.981**	**0.918**	**0.947**

Table 5: Qualitative Analysis Summary

Classifier	Analysis
Random Forest	Its precision (0.937) and recall (0.837) indicate reliable performance, although slightly lower than Linear SVM.
Logistic Regression	While the precision (0.850) is commendable, the recall (0.497) is relatively low, suggesting that the model may fail to identify certain high-priority requirements
Naive Bayes	The precision (0.227) and recall (0.333) indicate significant challenges in handling the prioritization task.
Linear SVM	The high precision (0.981) indicates its ability to accurately identify high-priority requirements, while its strong recall (0.918) reflects its effectiveness in retrieving most relevant requirements

Linear SVM emerged as the best-performing classifier, consistently outperforming others across all metrics. Random Forest also showed strong performance, making it a viable alternative for this task. The relatively lower performance of Logistic Regression and Naive Bayes underscores the importance of model selection in text-based prioritization tasks. These findings emphasize the potential of machine learning models, particularly Linear SVM, in leveraging modal verbs for accurate and efficient prioritization of functional requirements.

7. Conclusion and Future Work:

This study highlights the effectiveness and importance of modal verbs as one of the key features for requirement prioritization. Modal verbs serve as linguistic markers of obligation and importance, providing an interpretable and reliable foundation for mapping textual requirements to priority levels. By including their presence in requirement specifications, this approach highlights the potential of linguistic cues in enhancing the prioritization process.

In contrast to prior research and existing methods, which often emphasizes general keyword extraction or rule-based approaches, this study demonstrates the utility of combining modal verbs with machine learning classifiers. The extremely good performance of Linear SVM, which has achieved an accuracy of 96% and an F1-score of 0.947, supports the value of pairing robust feature extraction techniques, such as TF-IDF, with high-performing classification models. The findings validate the effectiveness of machine learning in automating and improving requirement prioritization.

Building on the results of this study, the research towards exploring the use of advanced NLP techniques for context-aware and semantic based prioritization seems to be open. This approach can enhance the understanding of contextual nuances in requirement specifications, especially when modal verbs alone may not suffice.

Acknowledgments

The authors would like to thank the post-graduate students who contributed their project requirements to this research, enabling a comprehensive and diverse dataset.

References

[1] Bukhsh, F. A., Bukhsh, Z. A., & Daneva, M. (2020). A systematic literature review on requirement prioritization techniques and their empirical evaluation. Computer Standards & Interfaces, 69, 103389.

[2] Hudaib, A., Masadeh, R., Qasem, M. H., & Alzaqebah, A. (2018). Requirements prioritization techniques comparison. Modern Applied Science, 12(2), 62.

[3] Kravchenko, T., Bogdanova, T., & Shevgunov, T. (2022, April). Ranking requirements using MoSCoW methodology in practice. In Computer Science On-line Conference (pp. 188-199). Cham: Springer International Publishing.

[4] Soufyane, A., Abdelhakim, B. A., & Ahmed, M. B. (2021, January). An intelligent chatbot using NLP and TF-IDF algorithm for text understanding applied to the medical field. In Emerging Trends in ICT for Sustainable Development: The Proceedings of NICE2020.

[5] Hujainah, F., Bakar, R. B. A., Nasser, A. B., Al-haimi, B., & Zamli, K. Z. (2021). SRPTackle: A semi-automated requirements prioritisation technique for scalable requirements of software system projects. Information and Software Technology, 131, 106501.

[6] Talele, P., & Phalnikar, R. (2021). Software requirements classification and prioritisation using machine learning. In Machine Learning for Predictive Analysis: Proceedings of ICTIS 2020 (pp. 257-267). Springer Singapore.

[7] Rahamathunnisa, U., Subhashini, P., Aancy, H. M., Meenakshi, S., & Boopathi, S. (2023). Solutions for Software Requirement Risks Using Artificial Intelligence Techniques. In Handbook of Research on Data Science and Cybersecurity Innovations in Industry .

[8] Kadhim, A. I. (2019). Survey on supervised machine learning techniques for automatic text classification. Artificial intelligence review, 52(1), 273-292.

[9] Usmonova, R. B. (2021). Semantic features of modal verbs in english grammar. Scientific progress, 1(4), 227-234.

[10] Naeem, M. Z., Rustam, F., Mehmood, A., Ashraf, I., & Choi, G. S. (2022). Classification of movie reviews using term frequency-inverse document frequency and optimized machine learning algorithms. PeerJ Computer Science, 8, e914.

[11] Uddin, S., Khan, A., Hossain, M. E., & Moni, M. A. (2019). Comparing different supervised machine learning algorithms for disease prediction. BMC medical informatics and decision making, 19(1), 1-16.

[12] Parmar, A., Katariya, R., & Patel, V. (2019). A review on random forest: An ensemble classifier. In International conference on intelligent data communication technologies and internet of things (ICICI) 2018 (pp. 758-763). Springer International Publishing.

[13] Montgomery, D. C., Peck, E. A., & Vining, G. G. (2021). Introduction to linear regression analysis. John Wiley & Sons.

[14] Salmi, N., & Rustam, Z. (2019, June). Naïve Bayes classifier models for predicting the colon cancer. In IOP conference series: materials science and engineering (Vol. 546, No. 5, p. 052068). IOP Publishing.

[15] Ghosh, S., Dasgupta, A., & Swetapadma, A. (2019, February). A study on support vector machine based linear and non-linear pattern classification. In 2019 International Conference on Intelligent Sustainable Systems (ICISS) (pp. 24-28). IEEE.

[16] Erickson, B. J., & Kitamura, F. (2021). Magician's corner: 9. Performance metrics for machine learning models. Radiology: Artificial Intelligence, 3(3), e200126.