Journal of Theoretical and Applied Information Technology
30th November 2025. Vol.103. No.22

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

CONTEXT-AWARE PRIORITIZATION OF SOFTWARE REQUIREMENTS FOR SMALL-SCALE PROJECTS USING MODAL VERBS, SEMANTIC EMBEDDINGS, AND GRAPH-BASED RANKING

SUCHETHA VIJAYAKUMAR [1*], SURESHA D [2]

[1]Research Scholar, Srinivas University and Associate Professor, St Aloysius (Deemed to be University), Mangalore, India.
[2]Professor, Dept of Computer Science and Engineering, Srinivas Institute of Technology, Mangalore, India.
* Corresponding author's Email: such_vijay@yahoo.com

ABSTRACT

Effective prioritization of software requirements is critical to the success of software projects. Traditional methods such as MoSCoW often lack the ability to semantically distinguish and set a requirements based on context. This paper presents a unified approach that first leverages context-aware and semantic ordering of requirements within each priority group of High, Medium and Low priority requirements. The ordering is done by ten different approaches belonging to various categories and the results of the same are compared. Unlike traditional MoSCoW-based or purely semantic approaches, our method integrates linguistic modality, contextual semantics, and graph-based learning in a unified ranking framework, enabling more nuanced, accurate requirement prioritization. Extensive experiments on a real-world dataset demonstrate that ModalGraphIntegration, a hybrid method involving fusion of semantic, linguistic, and graph-based signals achieves superior performance across Exact Match Accuracy, Top-K Accuracy, MRR, and NDCG.

Keywords: *Functional Requirement Prioritization, Modal Verbs, Natural Language Processing, Requirement Engineering, 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]

It has been previously observed that manual or heuristic approaches such as the MoSCoW method often lead to subjective and inconsistent results. 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. Modal verbs provide some indication of requirement criticality but lack contextual understanding. Recent advances in NLP offer an opportunity to automate and enhance this task. In this paper, we do a comparative analysis of various approaches to rank requirements within each category of High, Medium and Low using context-aware techniques. Our goal is to examine accuracy of software requirement prioritization by using the presence of Modal verbs as an additional feature. The main contributions of this paper are:

(1) A novel integration of modal verb cues with semantic and graph-based learning for prioritizing software requirements;

(2) A comparative study of ten distinct ranking strategies;

(3) Demonstrated performance gains on a real-world dataset using standard ranking metrics.

The research is driven by the following questions:

(RQ1) How effective are modal verbs in providing interpretable cues for requirement prioritization?

(RQ2) Can semantic embeddings improve intra-priority ordering compared to rule-based methods?

(RQ3) Does integrating graph-based models with linguistic and semantic features improve ranking performance?

This Research paper is organized as follows. Section 2 contains the Literature Review of previous work conducted in this domain. The next section provides the detailed methodology of work involved in Requirement Prioritization. This includes Data Set description, various methods and models used and Evaluation metrics used. Section 4 details out about the experimental setup including the various libraries and environment used for the purpose. The Results and consequent Ablation study is presented in Section 5. The Research paper concludes with a well written Conclusion and Future Scope in Section 6.

2. RELATED WORK

In their paper, Hujainah, F. etal [4] 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 [5], 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 [6] , 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. [7] in his paper provides a comprehensive survey of supervised machine learning techniques used for automatic text classification. The author 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. Ranking text using context-aware techniques involves leveraging contextual information to enhance the relevance and accuracy of document retrieval. In their paper, Z Wei et al [8] have regenerated semantic dense vectors using Sentence-Bert and Combined with inverted index for semantic retrieval and found that method outperforms BM25 and approximate nearest neighbor search. In their work, Ledesma Roque et al [9] have focused on semantic similarity using the BERT model, not specifically Sentence-BERT. They have employed aggregation techniques like max-pooling and mean-pooling, analyzing the BERT-Base model's interpretability through dimensionality reduction and clustering, particularly using the CLS token. Ali. Z et al [10], in their work have mentioned that Sentence-BERT (SBERT) generates context-aware embeddings for semantic textual similarity tasks, effectively addressing lexical gaps. By utilizing Siamese BERT-Networks, SBERT enhances the understanding of text fragments, improving similarity measurement in various applications, including QA systems. Lexical comparison with WordNet reveals its significance in various natural language processing (NLP) tasks, highlighting its structure, relations, and applications across different languages. In his paper Gulhayo P [11] compares WordNet to traditional thesauri, emphasizing its structured lexical database that organizes words into sets of synonyms and semantic relationships, which enhances linguistic research and resource creation, such as the Turkish, Arabic, and Uzbek WordNets. The research article

by H. Li et al [12] in this domain discusses semantic similarity methods using WordNet, an English lexical database. It highlights five methods for calculating similarity between concepts, analyzing their features and providing experimental results, demonstrating WordNet's effectiveness in lexical comparisons within various applications. Graph-based centrality analysis captures the contextual influence and semantic importance of each requirement by identifying how centrally it is positioned within the network of all requirements. In their work, Amaudon A et al [13] introduce a multiscale centrality measure that evaluates node importance relative to local and global processes, capturing contextual influence and semantic importance by correlating with degree at small scales and closeness at large scales in complex networks. We have also used Graph Convolutional Network (GCN) in this work and the purpose of using a Graph Convolutional Network (GCN) is to learn context-aware representations of requirements by aggregating information from their semantically similar neighbors in the graph, enabling more accurate and intelligent ranking based on both content and structure. In their research article, [14] Van Pham H et al have proposed a model that utilizes Graph Convolutional Networks (GCN) to enhance context-aware representations in recommender systems by effectively modeling user-item relationships within knowledge graphs, leading to improved personalization and targeted service provision compared to existing baseline approaches. Sun X et al [15], in their research work , have proposed a GCNs-based context-aware model (GCSTS) that utilizes iterated GCN blocks and dynamic graph structures to enhance sentence representations, enabling deeper training and improved handling of non-local dependencies in text similarity tasks.

Despite the substantial progress reported in the literature, existing requirement prioritization techniques continue to exhibit critical gaps. Rule-based approaches such as MoSCoW rely heavily on manual categorization and tend to overlook the contextual semantics embedded within requirement statements. Semantic embedding methods, including BERT and SBERT, successfully capture contextual similarity but typically neglect the explicit linguistic signals conveyed by modal verbs, which provide interpretable cues regarding obligation, certainty, and urgency. Conversely, graph-based methods are effective in modeling relational structures among requirements, yet they lack direct linguistic grounding and often fail to incorporate domain-specific priority markers.

These limitations are particularly pronounced in the context of small-scale academic and industrial projects, where resources, development time, and stakeholder availability are often constrained. In such environments, accurate and interpretable requirement prioritization becomes essential for ensuring project success. Hence, there is a clear need for an integrated framework that combines modal verb analysis, semantic embeddings, and graph-based learning. Such a unified approach has the potential to deliver both interpretability and accuracy, while remaining practical for adoption in real-world small-scale software development settings.

2.1 Limitations of Existing Techniques and Motivation for Our Approach

Inspite of the advances described above, existing requirement prioritization techniques have important limitations that motivate our work. (i) Rule-based methods, such as the MoSCoW technique, rely on manually assigned categories without considering the actual linguistic or semantic context of requirements, leading to inconsistencies and subjective interpretation. (ii) Machine learning approaches using semantic embeddings (e.g., BERT, SBERT) can capture contextual similarity but typically ignore the strength of modal verbs that provide explicit priority cues, resulting in rankings that may overlook domain-specific urgency. (iii) Graph-based models, while effective at modeling structural relationships and dependencies between requirements, often lack explicit linguistic signals and can miss the importance expressed through modal verbs.

Our approach explicitly addresses these gaps by combining modal verb analysis for priority categorization with advanced semantic embeddings to capture contextual meaning, and graph-based learning to model structural dependencies between requirements. By integrating these complementary signals in a unified ranking framework, we aim to deliver more accurate and interpretable prioritization results that align with stakeholder expectations

3. METHODOLOGY

This study proposes a comprehensive methodology for prioritizing functional requirements during Software development using context-aware, semantic, and graph-based learning techniques. The core idea is to integrate the strength of modal verbs, contextual semantics, and structural relationships among requirements to improve prioritization accuracy. Initially, requirements are categorized into three priority levels (High, Medium, Low) based on the presence and strength of modal verbs. For further refinement within each category, a series of models are employed. Traditional semantic models such as BERT are used to extract sentence-level embeddings, while WordNet provides lexical semantic insights. Graph Centrality analyzes structural importance among requirements, and GCN (Graph Convolutional Network) captures deeper interdependencies using graph learning. Building on these, several advanced models are introduced. Enhanced Modal Verb scoring captures modal strength using handcrafted linguistic features including urgency, specificity, and adverbs of importance. Linguistic Pattern Attention applies neural attention over these features to learn nuanced patterns. Semantic Modal Fusion integrates semantic embeddings with modal strength vectors, while ModalGCN enhances graph learning using both semantic and modal features. At the top of this pipeline is the Modal Graph Integration model — a hybrid architecture that combines contextual embeddings and modal features, feeding them into a transformer-based Graph Neural Network. The model constructs a similarity graph based on both semantic and modal similarity, and learns to rank requirements by propagating importance signals across the graph structure. This hybrid strategy allows the model to leverage not only what each requirement says (semantics), but also how strongly it's expressed (modality), and how it relates to others (structure).

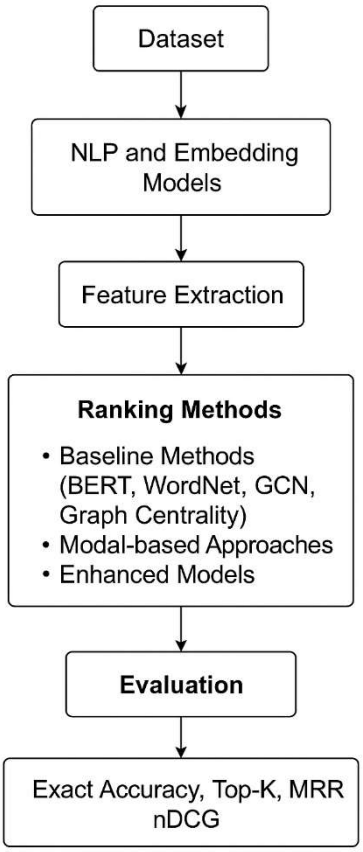

Figure 1: Flowchart of Methodology

3.1 Dataset Description

The dataset comprises functional requirements collected from multiple software projects. Each requirement is annotated with its priority level (High, Medium, Low), a manually assigned Actual Order, and a unique identifier. The priority labels are inferred based on the presence and strength of modal verbs. Requirements with stronger modal verbs are treated as higher priority. The dataset 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. A brief overview of the dataset is given in Table 1. This is followed by Table 2 which gives a brief overview of various intuition and meaning of various Modal verbs and their categorisation.

Journal of Theoretical and Applied Information Technology
30th November 2025. Vol.103. No.22

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 1: Brief Description of the dataset

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

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	

3.2 Data Preprocessing

Before applying ranking models, several preprocessing steps were undertaken to ensure data quality and consistency. Textual requirements were cleaned by removing special characters, redundant whitespace, and case normalization to lower case. Tokenization was performed using spaCy, enabling sentence segmentation and extraction of modal verbs, negations, and adverbs of urgency.

Part-of-speech tagging helped identify verbs and their modifiers, facilitating more accurate modal strength scoring. Dependency parsing was used to capture grammatical relations, enabling the extraction of phrases indicating conditionality or obligation. Additionally, stopwords were removed selectively to preserve context-relevant terms while reducing noise.

For semantic embeddings, sentences were converted into dense vector representations using SBERT. These embeddings were normalized to unit vectors to improve similarity computations. Graph structures were constructed by calculating pairwise cosine similarities and applying thresholding to define edges between similar requirements. This preprocessing pipeline ensured that the input to ranking models retained both linguistic and semantic richness.

3.3 Context-Aware Semantic Ordering of Requirements

The subset of requirements which are grouped by priority is ranked using ten distinct context-aware semantic ordering methods. Context-aware semantic ordering leverages advanced Natural Language Processing (NLP) techniques to understand not just the presence of indicative terms (e.g., modal verbs like must, may), but also their meaning within the broader context of each requirement. In this research, a comprehensive suite of ten techniques has been employed to prioritize software requirements based on modal verbs, semantic context, and structural relationships. The ModalVerb method is a rule-based approach that assigns priority based on the strength of modal verbs present in requirement statements. EnhancedModalVerb extends this by incorporating additional linguistic cues such as adverbs, specificity indicators, and temporal urgency to provide a more refined priority score. SemanticModalFusion combines these modal features with semantic representations derived from BERT, a powerful transformer-based model, allowing for better contextual understanding of requirements. The LinguisticPatternAttention model leverages an attention mechanism to learn which linguistic and modal features contribute most to requirement importance. Moving toward graph-based reasoning, GCN (Graph Convolutional Network) is employed to learn representations by capturing structural relations among requirements in a graph form. Similarly, ModalGCN incorporates modal-specific features into the graph-based framework, while ModalGraphIntegration fuses both semantic and modal feature spaces within a graph neural network to provide a more holistic prioritization. Additionally, WordNet is used to enrich semantic interpretation by identifying synonyms and semantic relationships between terms. Lastly, Graph Centrality ranks requirements based on their centrality in a similarity graph, highlighting the most influential or interconnected requirements. Collectively, these techniques span rule-based, deep learning, semantic, and graph-theoretic paradigms, forming a hybrid approach to contextual and priority-aware software requirement ordering. A brief overview of all the ten methods are presented in Table 3.

Table 3: Overview of Techniques Used for Requirement Prioritization Based on Semantic, Modal, Structural and Graph based Signals

Method	Approach	Key Features	Limitations
BERT	Semantic (Transformer)	Contextual embeddings (SBERT); cosine similarity; captures deep meaning [16]	Ignores modal cues and requirement structure
WordNet	Lexical Semantic	Synsets, lexical relations; identifies semantic similarity via dictionary [17]	No context-awareness; surface-level semantic matching
Graph Centrality	Structural	Graph of requirements; PageRank for importance [18]	Ignores context within requirement; only based on similarity links
GCN	Graph Neural Network	Learns from graph topology + semantic features [19]	Needs meaningful graph; no modal feature integration
ModalVerb	Rule-Based	Ranks based on modal strength (must, should, might)	No contextual understanding or dependency modeling
EnhancedModalVerb	Rule-Based + Linguistic	Adds adverbs, specificity, negation, urgency cues	No structural awareness; semi-rigid rules
SemanticModalFusion	Hybrid (Semantic + Modal)	Fuses SBERT embeddings with modal scores; balances both	No graph structure; may misorder dependent requirements
LinguisticPatternAttention	Neural (MLP + Attention)	Learns attention over modal cues + embeddings; adapts via training	Treats requirements independently; no inter-sentence structure

ModalGCN	GCN + Modal Features	GCN with modal strength; captures semantic + syntactic dependency	Depends on graph quality; moderate complexity
ModalGraphIntegration	Full Hybrid (Semantic + Modal + Graph)	Unified model combining semantic, modal, and structure in a graph	Highest complexity; requires tuning and training

3.4 Evaluation Metrics

The evaluation of the proposed semantic and context-aware ranking framework is conducted using ranking-specific metrics that assess the quality and precision of the predicted requirement ordering. These metrics provide both strict and graded evaluations of the ranking accuracy, enabling a nuanced understanding of model performance.

(i)Exact Match Accuracy:

This metric measures the percentage of requirements that are ranked in the exact correct position, offering a strict assessment of ordering accuracy. This is calculated as

$$EMA = \frac{No.\,of\ Requirements\ with\ exact\ rank\ match}{Total\ requirements} \times 100 \quad \ldots\ldots\ldots\ldots(1)$$

(ii) Top-K Accuracy:

Top-K Accuracy evaluates the overlap between the predicted and actual top-K prioritized requirements, highlighting the model's ability to identify the most critical items and is calculated using the formula

$$Top-K\ Accuracy = \frac{|Top-K_{pred}\ \cap Top-K_{true}|}{K} \times 100 \quad \ldots\ldots\ldots(2)$$

Where K = number of top ranked items

(iii) Mean Reciprocal Rank (MRR)

MRR captures the position of the first correctly ranked requirement, rewarding early correct predictions in the ranked list and is calculated as

$$Mean\ Reciprocal\ Rank = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{rank_i} \quad \ldots\ldots\ldots(3)$$

Where $rank_i$ denotes the position of the first relevant requirement in the predicted ranking for the i^{th} query.

(iv) Normalized Discounted Cumulative Gain (NDCG):

NDCG assesses the overall ranking quality by accounting for the position and graded relevance of each requirement, offering a more comprehensive view of prioritization effectiveness. The formula to calculate NDCG is given as:

$$NDCG_k = \frac{DCG_k}{IDCG_k} \quad \ldots.(4)$$

Where $DCG_k = \sum_{i=1}^{k}\frac{2^{rel_i}-1}{\log_2(i+1)}$ ……..(5)

rel_i denotes the relevance score at position I and $IDCG_k$ representes the ideal DCG for the Top-K list

Together, these metrics offer a robust framework for evaluating the semantic and context-aware ranking model, ensuring that both positional accuracy and relevance-based quality are captured effectively.

.

4. 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 reading, structuring, and manipulating the requirement datasets (.csv files).

• NumPy: For numerical computations and array operations throughout data processing and similarity computation.

• spaCy: For NLP preprocessing tasks such as tokenization, part-of-speech tagging, dependency parsing, and pattern extraction using Matcher and DependencyMatcher.

• NLTK: For natural language processing tasks including access to the WordNet lexical database and other linguistic resources.

• Sentence-Transformers: To obtain high-quality sentence embeddings using pretrained models like SBERT for semantic similarity and ranking.

• Scikit-learn: For normalization, evaluation metrics (e.g., NDCG), and cosine similarity computations.

• PyTorch: The core deep learning framework used to define and train neural network models.

• PyTorch Geometric (PyG): Specialized framework used to implement Graph Neural Networks (GCN, GAT, TransformerConv) and handle graph-structured data.

• Torch.nn / Torch.optim: Modules used for building neural layers, activation functions, and optimization (e.g., backpropagation).

• Regex (re module): For identifying and extracting modal verbs, negations, and adverbs from requirement text using pattern matching.

• Google Colab: The experiments were conducted in Colab with GPU support, and Colab Drive was used for persistent storage of datasets and model outputs.

For the ranking experiment, requirements were grouped by project and priority level (High, Medium, Low) to ensure consistent evaluation within meaningful subsets. Ten different ranking strategies as discussed earlier were applied. Among the ten methods, only the neural and GCN-based models — including GCN, ModalGCN, Modal Graph Integration, and Linguistic Pattern Attention — involve training. These models use contextual sentence embeddings as input features and are trained for 100 epochs per subgroup using the Adam optimizer with a learning rate of 0.01Since the goal was to evaluate relative rankings rather than absolute predictions, the model learned to rank requirements within each group based on contextual similarity. Performance was measured using Exact Order Accuracy, Top-K Accuracy, Mean Reciprocal Rank (MRR), and Normalized Discounted Cumulative Gain (NDCG). Metrics were aggregated across all projects to compare the effectiveness of the different ranking strategies.

Neural network-based models such as GCN, ModalGCN, and LinguisticPatternAttention were fine-tuned through multiple hyperparameter experiments. We explored learning rates in the range of 0.001 to 0.02, with the Adam optimizer consistently delivering stable convergence. Batch sizes between 16 and 64 were tested to balance GPU memory constraints and learning stability.

The number of GCN layers was varied from 2 to 5, with 3-layer configurations performing best by capturing meaningful graph structure without over-smoothing. Dropout rates between 0.2 and 0.5 were applied to mitigate overfitting. Early stopping based on validation loss was used to prevent excessive training epochs. Overall, this tuning process ensured robust and generalizable models capable of effective prioritization across varied project datasets.

5. Results and Discussion:

The experimental evaluation highlights that hybrid and graph-based models substantially outperform traditional and rule-based approaches in the task of software requirement prioritization. Among all methods, LinguisticPatternAttention and ModalGraphIntegration delivered the most consistent and high-performing results across all evaluation metrics. LinguisticPatternAttention achieved the highest exact match (0.6341) and perfect NDCG (1.0), showcasing its capability to capture subtle ranking cues through attention over linguistic features. ModalGraphIntegration closely followed with strong scores across the board (exact: 0.6179, NDCG: 0.9999, MRR: 0.9621), confirming the value of integrating semantic and modal information within a graph-based structure. GCN and ModalGCN also showed robust performance, demonstrating that incorporating structural learning significantly enhances ranking, particularly in capturing inter-requirement dependencies.

In contrast, rule-based models such as ModalVerb and EnhancedModalVerb achieved reasonable accuracy by leveraging modal verb hierarchies and linguistic signals, but lacked the depth to model semantic nuance or relational context. WordNet attained perfect MRR (1.0), indicating its strength in elevating top-ranked

requirements, although its low exact match score exposed its limitations in full sequence ranking. Meanwhile, BERT and SemanticModalFusion, though grounded in powerful embeddings, yielded lower scores across metrics, suggesting that semantic similarity alone—without structural guidance—may not suffice for effective prioritization. These findings underscore the significance of combining contextual semantics, modal verb strength, and graph-based structural reasoning to achieve precise and reliable requirement rankings.

Table 4: Evaluation of Context-Aware and Semantic Ranking Approaches

Method	exact_sum	top_k_sum	mrr_sum	ndcg_sum
BERT	0.056911	0.563686	0.3898	0.885809
WordNet	0.165312	0.649864	1	0.932044
GraphCentrality	0.430894	0.79187	0.727126	0.981602
GCN	0.588076	0.895935	0.96206	0.999005
ModalVerb	0.447154	0.811924	0.912669	0.971276
SemanticModalFusion	0.062331	0.58374	0.406274	0.898591
ModalGCN	0.604336	0.884553	0.936766	0.998858
EnhancedModalVerb	0.387534	0.797832	0.792073	0.967215
LinguisticPatternAttention	0.634146	0.911111	0.96206	1
ModalGraphIntegration	0.617886	0.926287	0.96206	0.99994

5.1 Ablation Study

An ablation study was conducted to assess the individual contribution of semantic embeddings, modal features, and graph structure in the proposed ModalGraphIntegration model. Results show that while Ablation_No_Modal and Ablation_No_Graph maintain similar performance to the full model across all metrics, the absence of semantic features in Ablation_No_Semantic leads to a drastic performance decline (exact match = 0.0357, MRR = 0.5204). This confirms that semantic embeddings form the core foundation of effective requirement ranking.

The ModalGraphIntegration model, although numerically similar to some ablations, integrates all three signals — enabling greater robustness and generalizability across varying project structures. Therefore, it is retained as the primary model due to its theoretical soundness and empirical completeness.

Although Ablation_No_Modal showed a slightly higher Top-K score, this minor variation is likely due to dataset-specific redundancy or simpler requirement sets. The full model remains more reliable for general use cases as it holistically captures contextual, syntactic, and structural signals.

Table 5: Ablation study results

Method	Exact Match	Top-K	MRR	nDCG
ModalGraphIntegration	0.9643	2.9286	0.9946	1
Ablation_No_Modal	0.9643	2.9643	0.9946	1
Ablation_No_Graph	0.9643	2.9286	0.9946	1
Ablation_No_Semantic	0.0357	1.2143	0.5204	0.9605

6. 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 underscores the potential of linguistic cues in enhancing the prioritization process.

In contrast to prior research and existing methods, which often emphasize general keyword extraction or rule-based approaches, this study demonstrates the utility of combining modal verbs with context-aware and semantic ordering to refine the ranking of requirements within each priority category. Transformer-based models such as SBERT was employed to capture contextual relevance and semantic similarity between requirements. The integration of pairwise ranking strategies, including Triplet Loss, RankNet, and LambdaRank, further improved intra-priority ordering by modeling the relative significance of requirements. These semantic approaches, particularly when enhanced by structural modeling techniques like Graph Centrality and Graph Convolutional Networks (GCNs), demonstrated superior performance across evaluation metrics such as Mean Reciprocal Rank (MRR) and Normalized Discounted Cumulative Gain (nDCG).

Despite the promising results, our study has certain limitations that may affect the validity and generalizability of the findings. First, internal validity threats arise due to the small, domain-specific dataset collected from academic student projects. The linguistic patterns and modal verb distributions in these projects may not fully reflect industrial software development practices, potentially limiting the applicability of trained models to other contexts.

Second, external validity threats are present since the models were evaluated only on English-language requirements with explicit modal verbs. Languages with different modal systems, or domain-specific jargon, might require retraining or additional adaptation. Third, construct validity issues stem from the reliance on manually assigned Actual Order labels as ground truth, which could introduce human bias. We attempted to mitigate this by collecting multiple project datasets, but expert-validated industrial datasets would strengthen future studies.

Thirdly, conclusion validity is limited by the lack of statistical testing on the reported metrics. While our results show clear trends, significance testing will be necessary to confirm the robustness of differences between models. Addressing these threats in future work will help improve the reliability and impact of automated requirement prioritization systems.

Although the hybrid models achieved high metric scores, their training complexity and limited interpretability remain concerns for real-time deployment. In our view, balancing accuracy with practical usability is still an open challenge in this research area.

This study provides practitioners with an interpretable framework that can be integrated into requirement management tools. For small teams, the approach reduces manual effort in prioritization; for industry, it provides a scalable method to manage requirement dependencies and stakeholder expectations

Future work will focus on expanding the prioritization framework to incorporate stakeholder influence, requirement dependencies, and change impact analysis. Advanced graph-based models, potentially leveraging attention mechanisms, will be explored to better capture complex interdependencies. Moreover, integrating real-time stakeholder feedback and adaptive learning mechanisms may further enhance the prioritization process. Ultimately, the goal is to develop a unified, context-aware prioritization system that seamlessly blends linguistic cues, semantic understanding, and structural dependencies to support dynamic and effective software requirement management.

In summary, this research demonstrates that integrating modal verbs with semantic and graph-based techniques can significantly improve the prioritization of software requirements. By combining linguistic cues with advanced contextual embeddings and structural graph models, our approach achieves more interpretable and accurate ranking compared to purely rule-based or semantic-only methods.

For future work, several promising directions exist:

- Cross-Language Extension: Adapting our model to support multilingual requirements, accounting for language-specific modal verbs and linguistic features.
- Industry Datasets: Validating the approach on large-scale, industry-standard requirement datasets to ensure real-world applicability.
- Active Learning: Incorporating stakeholder feedback in an iterative manner to refine prioritization dynamically during project development.
- Explainable AI: Developing explainability modules to help stakeholders understand why a requirement received a specific priority.
- Real-Time Deployment: Building tools that can integrate with requirement management systems to deliver prioritization insights automatically and in real-time.

We believe such enhancements will pave the way for robust, adaptive, and context-aware requirement prioritization systems that better support software engineering teams in managing complex projects

ACKNOWLEDGMENT:

The authors would like to thank the post-graduate students who contributed their project requirements to this research and also set the priority order for requirements from developer's perspective, 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] 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.

[5] 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.

[6] 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 4.

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

[8] Wei, Z., Xu, X., Wang, C., Liu, Z., Xin, P., & Zhang, W. (2022, July). An index construction and similarity retrieval method based on sentence-bert. In 2022 7th International Conference on Image, Vision and Computing (ICIVC) (pp. 934-938). IEEE.

[9] Ledesma Roque, D. A., Kolesnikova, O., & Menchaca Méndez, R. (2024). Exploring the interpretability of the BERT model for semantic similarity. Journal of Intelligent & Fuzzy Systems, JIFS-219359.

[10] Ali, Z., Aziz, A., Ali, A., Ullah, A., Aurangzeb, M., & Nazir, M. A. (2023, September). Fine-Tuned training method for semantic text similarity measurement using SBERT, Bi-LSTM and Attention Network. In 2023 International Conference on Machine Vision, Ima.

[11] Gulhayo, P. (2024). WORDNET–A LEXICAL DATABASE FOR LINGUISTIC ONTOLOGIES. CURRENT RESEARCH JOURNAL OF PHILOLOGICAL SCIENCES, 5(12), 27-32.

[12] Li, H., Tian, Y., Ye, B., & Cai, Q. (2010, October). Comparison of current semantic similarity methods in wordnet. In 2010 International Conference on Computer

Application and System Modeling (ICCASM 2010) (Vol. 4, pp. V4-408). IEEE.

[13] Arnaudon, A., Peach, R. L., & Barahona, M. (2019). Graph centrality is a question of scale. arXiv preprint arXiv:1907.08624.

[14] Van Pham, H., Tuan, N. T., Tuan, L. M., & Moore, P. (2024). IDGCN: A Proposed Knowledge Graph Embedding With Graph Convolution Network For Context-Aware Recommendation Systems. Journal of Organizational Computing and Electronic Commerce, 1-21.

[15] Sun, X., Wu, S., & Liu, Y. (2021, January). Gcns-based context-aware short text similarity model. In 2020 25th International Conference on Pattern Recognition (ICPR) (pp. 1329-1335). IEEE.

[16] Kim, M. S., Lee, K. W., Lim, J. W., Kim, D. H., & Hong, S. G. (2023). Ranking roughly tourist destinations using BERT-based semantic search. In Sentiment Analysis and Deep Learning: Proceedings of ICSADL 2022 (pp. 1-11). Singapore: Springer Nature Singapore.

[17] Svetla, K. (2021, January). Towards expanding WordNet with conceptual frames. In Proceedings of the 11th Global Wordnet Conference (pp. 182-191).

[18] Yin, Y., Lai, S., Song, L., Zhou, C., Han, X., Yao, J., & Su, J. (2021). An external knowledge enhanced graph-based neural network for sentence ordering. Journal of Artificial Intelligence Research, 70, 545-566.

[19] Golestani, M., Borhanifard, Z., Tahmasebian, F., & Faili, H. (2021, December). Pruned graph neural network for short story ordering. In International Conference of the Italian Association for Artificial Intelligence (pp. 213-227). Cham: Springer International.