Hybrid Feature Learning with TF-IDF and SBERT for Ambiguous Requirement Classification

Fariha Khalid[1], Muhammad Yaseen[2], Gohar Rahman[3]*,
Nauman Mazhar[4], Muhammad Asif Nauman[5], Aida Mustapha[6]
Department of Software Engineering, Riphah International University, Lahore, Pakistan [1, 2, 5]
Faculty of Computing and Informatics, Universiti Malaysia Sabah (UMS), 88400, Kota Kinabalu, Sabah Malaysia [3]
University of Central Punjab, Faculty of IT&CS, Lahore Pakistan [4]
Faculty of Computing, Universiti Malaysia Pahang Al-Sultan Abdullah, 26600 Pekan, Pahang, Malaysia [6]

***Abstract*—Ambiguity in Software Requirement Specifications (SRS) remains a major source of project delay, rework, and misinterpretation in software engineering. Traditional ambiguity detection approaches rely on lexical or rule-based techniques that capture surface-level patterns but fail to model contextual meaning. Recent transformer-based models improve semantic representation; however, when applied independently, they often overlook lexical ambiguity and remain sensitive to class imbalance. This study proposes a hybrid feature learning framework that integrates TF-IDF lexical representations with Sentence-BERT (SBERT) contextual embeddings for ambiguous requirement classification. The approach is evaluated on the Functional–Non-Functional Requirements (FR–NFR) dataset using Logistic Regression, Random Forest, and Support Vector Machine classifiers. Experimental results demonstrate that single-feature models produce unstable precision–recall trade-offs, particularly under severe class imbalance. In contrast, the proposed TF-IDF + SBERT hybrid representation consistently improves recall and F1-score. The best performance is achieved using Support Vector Machine, attaining an F1-score of 0.7122 and a recall of 0.6429, significantly outperforming standalone lexical and semantic baselines. The findings confirm that ambiguity detection is a multi-dimensional problem requiring both lexical frequency patterns and contextual semantic modelling. The proposed framework offers a reproducible and practically deployable solution for automated ambiguity detection in software requirements engineering.**

***Keywords*—Ambiguity detection; software requirements engineering; hybrid feature learning; TF-IDF; Sentence-BERT; Support Vector Machine**

I. Introduction

The software development process requires a clear statement of user requirements and expectations of the system. These requirements outline what the system does and its operational boundaries. They bridge the gap between the users and the developers to achieve the appropriate functionality. The requirements are normally expressed in a natural language to be easily understood [1]. This makes the requirements complex and causes errors in the development [2]. Requirements are considered the foundation of any software system, and the overall quality of the system largely depends on how effectively requirements are captured, analyzed, and implemented during the requirements engineering phase [3].

Requirements engineering is a major contributor to project success in software engineering. It is the stage where user requirements are gathered, analyzed, and recorded as a Software Requirements Specification (SRS). The SRS specifies what the system is expected to do. It specifies the functional and quality requirements of the system and serves as a formal contract between stakeholders and developers. A systematic and functional SRS minimizes ambiguity and serves as a foundation for the subsequent phases of design, implementation, and testing. In contrast, unusually imprecise or useless requirements tend to result in project delays, rework, extra cost, and software products of low quality. Thus, requirement engineering forms the basis of software systems that are reliable and maintainable [4]. Requirements engineering is a structured process that includes requirements elicitation, analysis, specification, validation, and management, all of which play a critical role in ensuring that software systems meet stakeholder expectations and maintain quality standards [5].

Ambiguity occurs when a requirement is open to more than one interpretation, which can perplex developers, testers, and clients. As a result, the delivered system may fail to meet user expectations. The ambiguous requirement affects the project cost, delivery time, and software quality, and it establishes communication barriers among these teams and the client [6] [7].

Traditional ambiguity detection techniques, such as rule-based systems and keyword-based searches, are limited to identifying surface-level ambiguity. These methods frequently generate a large number of false positives and false negatives because they do not capture the semantic and contextual meaning of requirement statements [8]. Domain-specific corpora have been shown to improve ambiguity resolution, yet generalizable solutions remain limited across heterogeneous requirement datasets [9]. Consequently, subtle ambiguity caused by vague expressions or implicit intent often remains undetected. In large-scale software systems, dependencies among requirements further complicate development, as improper handling of these dependencies can lead to project delays and reduced system reliability, highlighting the need for effective requirement analysis and prioritization mechanisms [10].

Machine learning approaches based on Bag-of-Words and TF-IDF representations have improved ambiguity detection by

*Corresponding author

learning statistical patterns from data. However, TF-IDF captures only term frequency information and ignores word order, sentence structure, and contextual meaning. As a result, these models perform poorly when ambiguity arises from semantic interpretation rather than lexical frequency, leading to low recall for ambiguous requirements [11].

Recent transformer-based models, including BERT, SBERT, and GPT, address some of these limitations by learning contextual and semantic relationships through large-scale pretraining [12]. While SBERT improves sentence-level semantic representation, when used independently, it may overlook lexical ambiguity caused by vague or domain-specific requirement terms. Early research has shown that ambiguity detection remains one of the most persistent challenges in requirements engineering due to inherent natural language variability and interpretation differences [13]. Moreover, transformer-based embeddings are often applied without effective integration with traditional lexical features, limiting their overall performance in ambiguity detection tasks within software requirements engineering.

This research aims to develop and empirically evaluate a context-aware framework for ambiguity detection in software requirement statements by integrating lexical and semantic feature representations within standard machine learning classifiers. Specifically, the study investigates the effectiveness of sentence-level semantic embeddings in distinguishing ambiguous and clear requirements, compares contextual representations such as SBERT with traditional TF-IDF-based lexical features, and examines the impact of hybrid feature learning on reducing false positives and false negatives under class imbalance. By systematically evaluating standalone and combined feature configurations using accuracy, precision, recall, and F1-score, the research seeks to determine whether integrating surface-level lexical signals with contextual semantic information leads to more stable, balanced, and reliable ambiguity detection. Through this hybrid feature learning strategy, the study aims to enhance automated ambiguity analysis in software requirements engineering and provide an empirically grounded, reproducible modelling framework for practical application.

The remainder of this study is organized as follows: Section II presents a background study of transformer-based and hybrid approaches in requirements engineering. Section III describes the research design, including the dataset, preprocessing pipeline, feature representations, classifiers, and evaluation metrics. Section IV reports the experimental results, followed by an analysis and discussion in Section V. Section VI concludes the study and outlines directions for future work.

II. Background Study

Transformer-based language models have significantly advanced natural language processing and have shown strong potential in software requirements engineering tasks. Unlike traditional lexical or rule-based techniques, transformer architectures rely on self-attention mechanisms that capture contextual dependencies between words across entire sentences. This allows them to interpret meaning based on the surrounding context rather than isolated tokens.

BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer) are among the most widely used transformer architectures in requirements engineering research. These models are pre-trained on large corpora and learn deep contextual representations that encode semantic relationships between words and phrases [14]. Their bidirectional attention mechanism enables them to detect subtle contextual shifts that often contribute to ambiguity in natural language requirements.

Moharil and Sharma [15] applied BERT-based contextual embeddings combined with clustering techniques to detect intra-domain and inter-domain ambiguity in software requirements. Their findings demonstrated that transformer-based representations could identify fine-grained semantic differences that traditional keyword-based approaches failed to detect. The model effectively captured contextual meaning across multiple application domains, highlighting the importance of contextual embeddings in ambiguity detection.

Further extensions of BERT have also been proposed for requirement classification tasks. Yucalar et al. [16] introduced a BERT-BiCNN model that integrates convolutional neural networks with BERT embeddings to capture sequential information and reduce feature dimensionality. Their approach outperformed classical machine learning baselines on benchmark datasets. Similarly, Kaur et al. [17] proposed PRCBERT, a prompt-based learning model designed to enhance zero-shot and supervised classification accuracy in requirements engineering datasets. These studies collectively demonstrate that transformer-based models consistently outperform traditional feature-based classifiers.

A comprehensive systematic review by Bahtiar et al. [18] analyzed 60 studies applying artificial intelligence techniques for requirement classification. The review concluded that transformer-based models, particularly those employing transfer learning, achieve higher accuracy and generalization performance compared to conventional machine learning and earlier deep learning methods. However, the review also highlighted a gap between academic experimentation and industrial adoption, noting that large-scale industrial validation remains limited.

Sentence-BERT (SBERT) has emerged as an adaptation of BERT designed to generate fixed-size sentence embeddings suitable for similarity and classification tasks [19]. In ambiguity detection, SBERT is particularly valuable because requirement ambiguity often depends on sentence-level meaning rather than isolated lexical cues. Moharil and Sharma [14] demonstrated that SBERT embeddings capture subtle semantic differences more effectively than TF-IDF or classical word embeddings. Additional studies combining SBERT with traditional classifiers such as Logistic Regression and SVM reported improved recall and F1-score in detecting ambiguous requirements compared to TF-IDF-based representations [12], [20]. Systematic mapping studies further confirm that hybrid and contextual representations remain underexplored in large-scale industrial ambiguity detection settings [21], [22].

Despite these advancements, transformer-based approaches exhibit several limitations. Many studies rely on relatively small or domain-specific datasets, limiting the generalizability of

results to industrial-scale requirements engineering contexts [23]. Transformer models also require substantial computational resources for fine-tuning and deployment, which may hinder practical implementation in real-world software development environments. Furthermore, although contextual embeddings improve semantic understanding, standalone transformer representations may overlook lexical-level ambiguity when terms are domain-specific or underrepresented in pretraining corpora.

Overall, transformer-based architectures represent the current state-of-the-art in ambiguity detection for software requirements. They provide superior contextual interpretation compared to earlier approaches. However, challenges related to scalability, industrial validation, domain adaptation, and integration with complementary lexical features indicate that further research is required to develop robust and generalizable ambiguity detection frameworks.

III. Research Design

This study adopts a structured experimental framework to evaluate lexical, semantic, and hybrid feature representations for ambiguous requirement classification using the FR–NFR dataset [24]. The design integrates controlled preprocessing, feature engineering, supervised learning, imbalance handling, and standardized evaluation to ensure methodological consistency and reproducibility. The overall research design workflow is illustrated in Fig. 1.

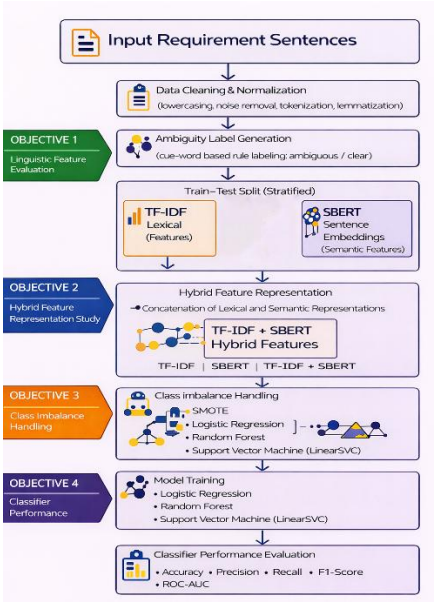

Fig. 1. Research design workflow.

A. Dataset and Experimental Setup

The FR–NFR requirements dataset contains 6008 requirement statements collected from publicly available software specification documents. For ambiguity detection, the dataset was transformed into a binary classification format where each requirement sentence s_i is associated with a label $y_i \in \{0,1\}$, such that 0 denotes a clear requirement and 1 denotes an ambiguous requirement. The dataset exhibits severe class imbalance: 5,665 statements (94.3%) are labelled as clear requirements and 343 statements (5.7%) are labelled as ambiguous, yielding a class imbalance ratio of approximately 16.5:1. This skewed distribution is a key challenge for minority-class detection and motivates the use of SMOTE oversampling. To ensure unbiased evaluation, the dataset was divided using stratified 80:20 train–test splitting, preserving the original class distribution in both subsets.

B. Text Preprocessing

All requirement statements were standardized before feature extraction. Text was converted to lowercase and cleaned by removing punctuation and non-alphabetic characters. Tokenization was performed to segment sentences into word units, followed by stopwords removal and lemmatization to reduce words to their base forms. This preprocessing pipeline was applied more aggressively for TF-IDF feature extraction to reduce noise, while minimally processed text was retained for SBERT to preserve contextual integrity.

C. Feature Representation

TF-IDF represents term importance relative to the corpus. For a term t in document d, term frequency is defined as:

$$TF\ (t,d)\ = \frac{count\ (t,d)}{total\ terms\ in\ d} \quad (1)$$

Inverse document frequency is computed as:

$$IDF(t)\ =\ log\ (\frac{N}{1+df\ (t)}) \quad (2)$$

where, N is the total number of documents and df(t) is the number of documents containing term t. The TF-IDF weight is then calculated as:

$$TF-IDF = TF(t,d) \times IDF(t) \quad (3)$$

Unigram, bigram, and trigram representations were extracted to capture both single-word and short phrase ambiguity signals.

To model contextual semantics, Sentence-BERT (SBERT) was employed using the pretrained *all-MiniLM-L6-v2* model. This model produces 384-dimensional sentence embeddings using a mean pooling strategy over the final transformer layer token outputs, followed by L2 normalization. The all-MiniLM-L6-v2 variant was selected for its balance between computational efficiency and semantic representation quality. Each requirement sentence s is encoded into a dense embedding:

$$v_{SBERT} \in \mathbb{R}^{384} \quad (4)$$

This embedding captures sentence-level meaning through transformer-based contextual representation.

For the hybrid configuration, lexical and semantic representations were combined. Let $x_{tfidf}(s) \in \mathbb{R}^m$ and $x_{sbert}(s) \in \mathbb{R}^{384}$. After L2 normalization:

$$\hat{x} = \frac{x}{\|x\|2} \quad (5)$$

The hybrid vector is constructed as:

$$xhybrid(s) = \hat{x}tfidf\ \oplus\ \hat{x}sbert \quad (6)$$

where, $\oplus$ denotes concatenation. The resulting representation integrates surface-level lexical frequency patterns

with contextual semantic information. Fig. 1 illustrated the complete hybrid feature construction pipeline, showing the parallel extraction of lexical (TF-IDF) and semantic (SBERT) features, followed by normalization and feature concatenation before classification.

D. Class Imbalance Handling

To address minority-class underrepresentation, the Synthetic Minority Over-sampling Technique (SMOTE) was applied to the training data only. Similar imbalance challenges have been reported in defect prediction and requirements classification research, where minority-class detection significantly affects recall-oriented performance [25]. Synthetic samples are generated according to:

$$Xnew = xi + \lambda(xnn - xi)$$

where, xi is a minority instance, x_{nn} is one of its nearest neighbors, and $\lambda \in (0,1)$. This strategy mitigates bias toward the majority class while preventing information leakage from the test set.

E. Classification Models

$$P(y = 1 \mid x) = \frac{1}{1+e-(w \cdot x+b)} \quad (7)$$

where, classification is determined using a threshold of 0.5. Random Forest constructs multiple decision trees and predicts the final class through majority voting:

$$y\hat{} = mode\{T1(H), T2(H), \ldots, Tn(H)\} \quad (8)$$

Support Vector Machine with a linear kernel learns a separating hyperplane defined as:

$$f(x) = w \cdot x + b \quad (9)$$

and assigns class labels based on the sign of f(x). The use of a linear kernel is appropriate due to the high-dimensional nature of TF-IDF and hybrid feature spaces. Algorithm 1 presents the hybrid TF-IDF and SBERT ambiguity detection framework.

Algorithm 1: Hybrid TF-IDF and SBERT Ambiguity Detection Framework

The proposed hybrid ambiguity detection framework can be formally described as follows.

Input: Requirement sentence s

Output: Predicted label y ∈ {0, 1}

Initialize pretrained SBERT encoder

Initialize TF-IDF vectorizer

Initialize classifier $\boldsymbol{f}(\cdot) \in \{\boldsymbol{LR}, \boldsymbol{RF}, \boldsymbol{SVM}\}$

Preprocess sentence s

Convert to lowercase

Remove punctuation and non-alphabetic characters

Tokenize and lemmatize

Compute lexical feature vector $x_{tfidf}(s)$

Compute semantic embedding xsbert(s) ∈ R^{384}

$$\widehat{x_{tfidf}} = \frac{x_{tfidf}}{\|x_{tfidf}\|2} \quad (10)$$

$$\widehat{x_{SBERT}} = \frac{x_{SBERT}}{\|x_{SBERT}\|2} \quad (11)$$

Construct hybrid representation

$$xhybrid(s) = \hat{x}tfidf \oplus \hat{x}sbert$$

If training phase then

Apply SMOTE on hybrid training vectors

Train classifier f (x_{hybrid})

End

Compute prediction score

$$\boldsymbol{score = f(xhybrid)}$$

If score ≥ 0 then

y^=1

Else

y^=0

End

Return y^

F. Evaluation Metrics

Model performance was assessed using standard classification metrics. Accuracy is defined as:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (12)$$

Precision and recall are computed as:

$$Precision = \frac{TP}{TP+FP}, Recall = \frac{TP}{TP+FN} \quad (13)$$

The F1-score, representing the harmonic mean of precision and recall, is calculated as:

$$F1 - score = \frac{2*Recall*precision}{Recall+precision} \quad (14)$$

Overall, the research design systematically evaluates lexical, semantic, and hybrid feature representations under controlled experimental conditions, ensuring that performance differences arise from feature learning strategies rather than inconsistencies in preprocessing, data splitting, or model configuration.

G. Computational Overhead and Deployment Considerations

From a practical deployment perspective, the hybrid framework involves two stages of feature extraction with distinct computational profiles. TF-IDF vectorization is lightweight, requiring only term-frequency counting over the vocabulary, and scales linearly with corpus size. SBERT inference using the all-MiniLM-L6-v2 model requires a transformer forward pass per sentence; on a standard CPU, average inference latency is approximately 5–10 milliseconds per sentence, making it feasible for batch processing in software development tool chains. The SVM classifier, operating on the concatenated high-dimensional feature vector, adds negligible inference overhead. The total pipeline can be integrated into requirements engineering tools as an offline batch analyzer or as

a lightweight service for interactive requirement review, with memory requirements well within the constraints of modern development environments.

IV. Results

This section presents the experimental evaluation of the proposed ambiguity detection framework on the Functional–Non-Functional Requirements (FR–NFR) dataset. The dataset contains 6008 software requirement statements, out of which 5665 are labelled as clear and 343 as ambiguous, indicating a severe class imbalance. All models were evaluated using accuracy, precision, recall, F1-score, and overall score. Particular emphasis is placed on recall and F1-score because ambiguity detection is inherently an imbalanced classification problem.

Three classical machine-learning classifiers were selected: Logistic Regression (LR), Random Forest (RF), and Support Vector Machine (SVM). Each classifier was evaluated using three feature configurations: TF-IDF, SBERT, and the hybrid TF-IDF + SBERT representation. The purpose of this evaluation is to systematically analyze how lexical and semantic representations influence ambiguity detection under consistent modelling conditions.

A. Baseline TF-IDF Results

Baseline experiments were conducted using TF-IDF lexical features. The results show that TF-IDF-based models achieve moderate to high accuracy; however, their effectiveness in detecting ambiguous requirements remains limited. Logistic Regression achieved an accuracy of 0.760355 with a recall of 0.584416, but precision was very low (0.133531), resulting in a weak F1-score of 0.217391. Similarly, SVM with TF-IDF obtained a recall of 0.629870, but precision dropped to 0.120497, producing an F1-score of only 0.202294. Random Forest achieved higher accuracy (0.925666), but recall remained moderate (0.454545), indicating that many ambiguous requirements were still missed.

These findings demonstrate that lexical frequency features alone are insufficient for reliable ambiguity detection. Although TF-IDF captures word importance, it does not model contextual relationships between words, leading to unstable precision–recall trade-offs under class imbalance.

B. SBERT Results

To evaluate the impact of contextual semantic representation, SBERT embeddings were used with LR, RF, and SVM classifiers. SBERT-based models showed improved contextual understanding compared to TF-IDF-only baselines. Logistic Regression achieved an F1-score of 0.455100 with a recall of 0.493500, while SVM achieved an F1-score of 0.445100 with a recall of 0.461000. These values indicate better balance compared to lexical-only models.

However, Random Forest trained on SBERT embeddings achieved very high accuracy (0.953800) and precision (0.837200), but recall dropped to 0.233800, producing an F1-score of 0.365500. This behavior indicates a strong bias toward the majority class, where ambiguous requirements are rarely predicted.

These results confirm that although SBERT captures contextual meaning and sentence-level semantics, semantic embeddings alone do not provide stable ambiguity detection under severe class imbalance.

C. Hybrid TF-IDF + SBERT Results

The proposed hybrid framework integrates lexical TF-IDF features with semantic SBERT embeddings into a unified representation. The objective of this design is to capture complementary information that cannot be effectively modelled using single-feature configurations. All hybrid models were evaluated using LR, RF, and SVM.

The experimental results demonstrate a clear and consistent improvement when hybrid features are applied. SVM with TF-IDF + SBERT achieved the best overall performance, obtaining an accuracy of 0.970400, precision of 0.798400, recall of 0.642900, and an F1-score of 0.712200, resulting in the highest overall score of 0.780975. Logistic Regression with hybrid features also demonstrated strong performance, achieving an F1-score of 0.684000. Random Forest benefited from hybrid features in terms of precision (0.833300), but recall remained low (0.259700), limiting its F1-score.

The improvement is primarily observed in recall and F1-score, indicating better detection of ambiguous requirements. These results confirm that ambiguity detection requires simultaneous modelling of lexical frequency patterns and contextual semantic information. The full comparison of all classifiers and feature configuration combinations is presented in Table I.

TABLE I. Performance Comparison on FR–NFR Dataset

Feature Set	Model	Accuracy	Precision	Recall	F1-Score	Overall Score
TF-IDF	SVM	0.717086	0.120497	0.629870	0.202294	0.417437
TF-IDF	LR	0.760355	0.133531	0.584416	0.217391	0.423923
TF-IDF	RF	0.925666	0.374332	0.454545	0.410557	0.541275
SBERT	RF	0.953800	0.837200	0.233800	0.365500	0.597575
SBERT	LR	0.932700	0.422200	0.493500	0.455100	0.575875
SBERT	SVM	0.934500	0.430300	0.461000	0.445100	0.567725
TF-IDF + SBERT	SVM	0.970400	0.798400	0.642900	0.712200	0.780975
TF-IDF + SBERT	LR	0.968600	0.800000	0.597400	0.684000	0.762500

TF-IDF + SBERT	RF	0.954900	0.833300	0.259700	0.396000	0.610975

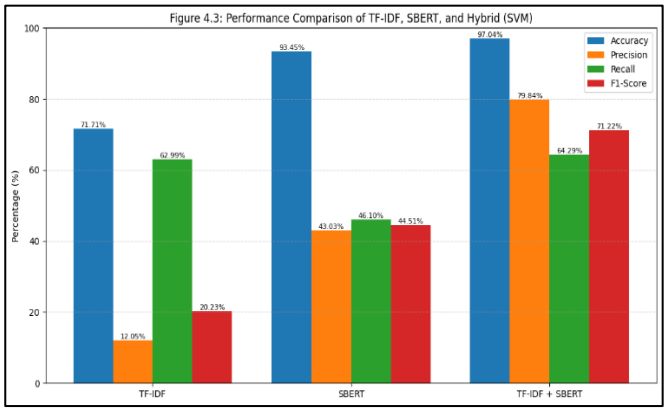

Fig. 2. Performance comparison of TF-IDF, SBERT, and a hybrid model.

V. Analysis and Discussion

The experimental results reveal significant differences between single-feature and hybrid feature representations. In the FR–NFR dataset, class imbalance strongly influences model behavior. Baseline TF-IDF models achieve moderate to high accuracy due to the dominance of clear requirements, yet they fail to consistently detect ambiguous requirements, as reflected in low precision and F1-scores.

SBERT embeddings improve contextual understanding and produce more balanced precision–recall behavior compared to TF-IDF. However, when used independently, SBERT remains sensitive to class imbalance, particularly in tree-based models such as Random Forest, which show conservative prediction behavior. Previous studies have similarly observed that contextual embeddings enhance requirement classification but require integration with complementary features for stable minority-class detection [15], [19].

The hybrid TF-IDF + SBERT representation consistently improves recall and F1-score across classifiers. This confirms that ambiguity is a multi-dimensional phenomenon requiring both lexical and semantic modelling. Lexical features capture frequency-based patterns associated with vague terms, while SBERT embeddings encode contextual meaning and sentence intent. Their integration reduces false negatives and produces more stable classification boundaries.

Support Vector Machine consistently achieves the best performance under a hybrid configuration. Its margin-based learning mechanism allows effective separation within the high-dimensional fused feature space. Unlike Logistic Regression, which tends to favor the majority class, and Random Forest, which remains conservative, SVM maintains a better balance between precision and recall.

Overall, the results demonstrate that hybrid feature integration provides a robust and generalizable solution for ambiguity detection in functional–non-functional requirements. The consistent improvement in recall and F1-score confirms the practical utility of the proposed framework, particularly when combined with Support Vector Machine. The cross-classifier performance trends are further visualized in Fig. 2, which illustrates the F1-score comparison across all feature configurations and classifiers. The obtained F1-score reflects the inherent difficulty of ambiguity detection under severe class imbalance, where minority-class instances are limited and context-dependent.

A key limitation of the current evaluation is that the framework is assessed on a single dataset from one source. Without cross-dataset validation or transfer experiments on requirements from different application domains, it remains uncertain whether the observed performance generalizes beyond the FR–NFR dataset. Future work should evaluate the framework on additional publicly available requirements corpora to establish broader generalizability. Furthermore, while the hybrid approach outperforms standalone TF-IDF and SBERT baselines, a direct comparison against fine-tuned transformer models such as BERT or RoBERTa on the same dataset would provide stronger evidence of the framework's competitive standing relative to the current state-of-the-art [23]. Due to computational constraints and the focus on feature-level hybrid representation, fine-tuned transformer baselines such as BERT and RoBERTa were not included in the current experimental evaluation.

VI. Conclusion and Future Work

This study investigated ambiguity detection in software requirements using a hybrid feature learning framework that integrates TF-IDF and SBERT representations within classical machine learning classifiers. Ambiguous requirements remain a major source of project delay and miscommunication, and traditional lexical models are often insufficient to capture context-dependent ambiguity.

The experimental results on the FR–NFR dataset show that single-feature models such as TF-IDF or SBERT, when used independently, produce unstable recall and limited F1-scores under severe class imbalance. TF-IDF captures lexical importance but lacks contextual understanding, while SBERT provides semantic awareness but struggles when applied alone. These findings confirm that ambiguity is multi-dimensional and cannot be reliably detected through isolated feature representations.

The hybrid TF-IDF + SBERT configuration consistently achieved the strongest and most balanced performance, particularly when evaluated using Support Vector Machine. Improvements were most evident in recall and F1-score, indicating a reduction in false negatives, which is critical in ambiguity detection tasks. The results demonstrate that integrating lexical and semantic cues within a unified representation significantly enhances classification robustness and minority-class detection.

Overall, the proposed hybrid framework provides a reproducible and empirically validated solution for ambiguity detection, outperforming single-feature baselines and offering practical applicability in requirements engineering.

A. Limitations

The study is limited to English-language requirements and focuses on sentence-level ambiguity without modelling document-level context. The framework is evaluated on a single dataset from one source; cross-dataset validation across requirements from different application domains has not been performed, which limits conclusions about generalizability. Although the FR–NFR dataset is substantial, it may not fully represent the linguistic diversity of large industrial specifications. The proposed hybrid approach is not directly compared against fine-tuned transformer-based baselines such as BERT or RoBERTa on the same dataset, which would be necessary to position the framework conclusively relative to the current state-of-the-art. In addition, SBERT embeddings were not domain-adapted to software engineering corpora.

B. Future Work

Future work may extend this framework to larger and multilingual datasets to improve generalizability, and should include cross-dataset validation on requirements from diverse application domains. A direct comparison against fine-tuned transformer-based models such as BERT and RoBERTa on the FR–NFR dataset would clarify the competitive standing of the hybrid approach relative to the current state-of-the-art [23]. Domain-specific fine-tuning of transformer embeddings could further enhance contextual sensitivity. Incorporating explainable AI techniques such as SHAP or LIME would improve transparency and stakeholder trust. Finally, document-level modelling approaches could enable detection of cross-requirement ambiguity, strengthening real-world applicability.

REFERENCES

[1] A. Birhane, "The Impossibility of Automating Ambiguity," Artif. Life, vol. 27, no. 1, pp. 44–61, 2021, doi: 10.1162/artl_a_00336.

[2] M. Q. Riaz, W. H. Butt, and S. Rehman, "Automatic Detection of Ambiguous Software Requirements: An Insight," in 2019 5th International Conference on Information Management (ICIM), 2019, pp. 1–6. doi: 10.1109/INFOMAN.2019.8714682.

[3] M. Yaseen, A. Mustapha, M. A. Shah, and N. Ibrahim, "A hybrid technique using minimal spanning tree and analytic hierarchical process to prioritize functional requirements for parallel software development," Requir. Eng., vol. 28, no. 3, pp. 347–376, Feb. 2023, doi: 10.1007/s00766-023-00397-9.

[4] O. Kosenkov et al., "Systematic Mapping Study on Requirements Engineering for Regulatory Compliance of Software Systems," 2024. doi: 10.48550/arXiv.2411.01940.

[5] M. Yaseen and Z. Karamat, "Requirements Engineering Model (REM): An Assessment Model for Software Vendor Organizations," J. Softw. Evol. Process, vol. 37, no. 4, Apr. 2025, doi: 10.1002/smr.70020.

[6] A. Veizaga, S. Y. Shin, and L. C. Briand, "Automated Smell Detection and Recommendation in Natural Language Requirements," IEEE Trans. Softw. Eng., vol. 50, no. 4, pp. 695–720, 2024, doi: 10.1109/TSE.2024.3361033.

[7] S. K. Aqsa Rasheed, Bushra Zafar, Tehmina Shehryar, Naila Aiman Aslam, Muhammad Sajid, Nouman Ali, Saadat Hanif Dar, "Requirement Engineering Challenges in Agile Software Development," 2021.

[8] S. Ezzini, S. Abualhaija, C. Arora, and M. Sabetzadeh, "Automated handling of anaphoric ambiguity in requirements: a multi-solution study," in Proceedings of the 44th International Conference on Software Engineering, in ICSE '22. New York, NY, USA: Association for Computing Machinery, 2022, pp. 187–199. doi: 10.1145/3510003.3510157.

[9] S. Ezzini, S. Abualhaija, C. Arora, M. Sabetzadeh, and L. C. Briand, "Using Domain-Specific Corpora for Improved Handling of Ambiguity in Requirements," in 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), 2021, pp. 1485–1497. doi: 10.1109/ICSE43902.2021.00133.

[10] M. Yaseen, A. Bahar, G. Rahman, M. Ashurov, and E. Kholiyarov, "Minimizing Inter-Dependencies in Functional Requirements for Timely Delivery of Software Projects: A Prioritization Approach Using AHP and Spanning Trees," J. Softw. Evol. Process, vol. 37, no. 11, p. e70063, 2025, doi: https://doi.org/10.1002/smr.70063.

[11] Y. Liu, A. Medlar, and D. Głowacka, "Lexical ambiguity detection in professional discourse," Inf. Process. Manag., vol. 59, no. 5, p. 103000, 2022, doi: https://doi.org/10.1016/j.ipm.2022.103000.

[12] A. Ferrari et al., "Detecting requirements defects with NLP patterns: an industrial experience in the railway domain," Empir. Softw. Eng., vol. 23, no. 6, pp. 3684–3733, 2018, doi: 10.1007/s10664-018-9596-7.

[13] B. Gleich, O. Creighton, and L. Kof, "Ambiguity Detection: Towards a Tool Explaining Ambiguity Sources," in Requirements Engineering: Foundation for Software Quality, R. Wieringa and A. Persson, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 218–232.

[14] M. Alhamed and T. Storer, "Evaluation of Context-Aware Language Models and Experts for Effort Estimation of Software Maintenance Issues," in 2022 IEEE International Conference on Software Maintenance and Evolution (ICSME), 2022, pp. 129–138. doi: 10.1109/ICSME55016.2022.00020.

[15] A. Moharil and A. Sharma, "Identification of intra-domain ambiguity using transformer-based machine learning," in Proceedings of the 1st International Workshop on Natural Language-based Software Engineering, New York, NY, USA: ACM, May 2022, pp. 51–58. doi: 10.1145/3528588.3528651.

[16] F. Yucalar, "Developing an Advanced Software Requirements Classification Model Using BERT: An Empirical Evaluation Study on Newly Generated Turkish Data," Appl. Sci., vol. 13, no. 20, 2023, doi: 10.3390/app132011127.

[17] K. Kaur and P. Kaur, "Improving BERT model for requirements classification by bidirectional LSTM-CNN deep model," Comput. Electr. Eng., vol. 108, p. 108699, 2023, doi: https://doi.org/10.1016/j.compeleceng.2023.108699.

[18] S. A. H. Bahtiar, I. Universitas Islam Indonesia, C. K. Dewa, I. Universitas Islam Indonesia, A. Luthfi, and I. Universitas Islam Indonesia, "Comparison of Naïve Bayes and Logistic Regression in Sentiment Analysis on Marketplace Reviews Using Rating-Based Labeling," 2023.

[19] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," CoRR, vol. abs/1908.1, 2019.

[20] N. Rankovic and D. Rankovic, "Labeling issues through semantic patterns in open-source Agile practices," Knowledge-Based Syst., vol. 328, p. 114197, 2025, doi: https://doi.org/10.1016/j.knosys.2025.114197.

[21] L. Zhao et al., "Natural Language Processing for Requirements Engineering: A Systematic Mapping Study," ACM Comput. Surv., vol. 54, no. 3, Apr. 2021, doi: 10.1145/3444689.

[22] R. Sonbol, G. Rebdawi, and N. Ghneim, "The Use of NLP-Based Text Representation Techniques to Support Requirement Engineering Tasks: A Systematic Mapping Review," IEEE Access, vol. 10, pp. 62811–62830, 2022, doi: 10.1109/ACCESS.2022.3182372.

[23] A. Ferrari and A. Esuli, "An NLP approach for cross-domain ambiguity detection in requirements engineering," Autom. Softw. Eng., vol. 26, pp. 559–598, 2019, doi: 10.1007/s10515-019-00261-7.

[24] S. S. T. Sonali, "FR-NFR Dataset," 2024, [Online]. Available: https://data.mendeley.com/datasets/4ysx9fyzv4/1/files/15b2b7dd-15d3-4db1-a510-1011fb1465a1

[25] G. Giray, K. E. Bennin, Ö. Köksal, Ö. Babur, and B. Tekinerdogan, "On the use of deep learning in software defect prediction," J. Syst. Softw., vol. 195, p. 111537, 2023, doi: https://doi.org/10.1016/j.jss.2022.111537.