Feature Engineering for Agile Requirement Management Using Semantic Analysis

Abstract

Efficient management and prioritization of software requirements are critical challenges in agile projects, where requirements constantly evolve due to changing user needs, business goals, and regulatory updates. This paper explores the role of semantic feature extraction in enabling adaptive management strategies. Using the PROMISE Expanded Dataset and the Coquina Dataset, we employed TF-IDF weighted Word2Vec for advanced tokenization and feature extraction.Latent Dirichlet Allocation (LDA) was used to analyze how preprocessing steps like stop word removal impact topic representation, revealing that removing stop words improved topic specificity and coherence. To address class imbalance, Synthetic Minority Over-sampling Technique (SMOTE) was applied, enhancing the model's ability to handle underrepresented classes effectively.Principal Component Analysis (PCA) reduced the dimensionality of TF-IDF weighted Word2Vec embeddings from 100 features to 30, while Analysis of Variance (ANOVA) identified the most significant features for classification. The results obtained identifiedthree features to have p-values below 0.05 as statistically significant, p-value = 0.0000605, p-value = 0.00000000469, and p-value = 0.0024.These extracted features could be used as input to training machine learning models for predicting and managing software requirements adaptively during agile development.With the reduction of ambiguities and sentiments of the user at the requirement phase, the development phase could be undertaken seamlessly with ease.

1.0 Introduction

Effective requirement management and prioritization are fundamental to agile software development. Agile methodologies emphasize iterative progress and flexibility, making it essential to have robust systems for handling requirements. This paper explores the use of semantic analysis and adaptive management strategies to improve the classification and prioritization of requirements. We leverage the PROMISE Expanded Dataset and the Coquina Dataset to demonstrate our approach.The main objective of this research is to highlight the pivotal role of semantic feature extraction in enabling adaptive management strategies within machine learning models for agile software development. This research focuses on enhancing the accuracy and adaptability of managing changing software requirements by leveraging advanced feature extraction techniques. The specific objectives are to:

a. Utilize historical data from the PROMISE_exp repository and past software projects from Coquina Software Company Limited to identify patterns and trends related to requirement changes in agile environments.
b. Carry out Data explorationof the datasets.
c. Extract some features of the datasets using extraction techniques, including TF-IDF weighted Word2Vec and Latent Dirichlet Allocation (LDA), to improve the representation and analysis of requirement text.
d. Perform feature selection on the datasets.

2.0 Literature Review

In recent years, the development of adaptive management strategies foragile requirements has gained increasing attention, with a particularfocus on semantic feature extraction. This domain encompasses avariety of natural language processing (NLP) techniques, such as stopword removal, Latent Dirichlet Allocation (LDA), sentiment analysis,and various text representation methods including Bag of Words,TF-IDF, Word2Vec, and embeddings. This literature review integratesand synthesizes recent research findings to provide a comprehensiveunderstanding of these techniques, identify existing knowledge gaps,and propose future research directions.

A foundational NLP technique is stop word removal, which involveseliminating commonly used words (e.g., "the," "is," "and") to focus onmore meaningful content. Research by Abbas et al. (2021) explores therelationship between requirements similarity and software similarity,emphasizing the importance of effective feature extraction techniqueslike stop word removal in understanding requirements within agileenvironments. The removal of stop words improves the quality offeature engineering, allowing for more insightful semantic analysis.

Latent Dirichlet Allocation (LDA) has emerged as a powerful tool foruncovering hidden thematic structures within a corpus. Bing and Xiuwen(2022) highlight the potential of deep learning models such as NFRNetfor classifying non-functional requirements (NFRs), suggesting the useof LDA and word embeddings to enhance feature extraction in agilerequirements management. Similarly, Asad and Muqeem (2022) proposed that advanced feature engineering methods like LDA can better capturethe dynamic nature of requirements in agile development.

Sentiment analysis, a technique used to extract subjective informationfrom text, is also relevant in the context of adaptive managementstrategies. Franch et al. (2020) present a vision for data-drivenrequirements engineering, emphasizing the need for integratingsentiment analysis with feature engineering to optimize the managementof requirements. The application of sentiment analysis offers deeperinsights into user feedback and prioritization, particularly in agilesettings where requirements frequently evolve.

To represent text data effectively, Bag of Words and TF-IDF arefundamental techniques in feature extraction. Canedo and Mendes (2020)provide a comparative study of machine learning algorithms forclassifying software requirements, highlighting TF-IDF's effectivenessin managing agile requirements. The TF-IDF model emphasizes theimportance of specific terms within a document, which is critical forfeature extraction in requirement management.

Word2Vec, a model designed to generate word embeddings that capturesemantic meaning, complements these text representation techniques.Research by Kurtanović and Maalej (2017) demonstrates the potential ofmachine learning models, such as Word2Vec, in classifying functionaland non-functional requirements. The integration of Word2Vec withTF-IDF balances semantic context with term importance, enhancingfeature extraction in agile requirements management.

Deep learning methods continue to advance feature extractiontechniques. Navarro-Almanza et al. (2017) propose the use of deeplearning for software requirements classification, reinforcing therole of semantic analysis and feature engineering in improving theaccuracy of agile

requirement management. Yang et al. (2023) furtherexplored clustering user stories in agile development, showcasing howsemantic analysis can enhance the categorization and management of dynamic requirements.

This exploration of NLP techniques, deep learning models, and theirapplications in adaptive management strategies underscores theimportance of continued research and development in this evolvingfield. By integrating these methods, researchers and practitioners canimprove the management of agile requirements, addressing challengesposed by the dynamic nature of software development environments.

3.0Research Methodology

The methodology involves preliminary data processing with topic modeling to evaluate the impact of stop word removal, followed by feature extraction using weighted TF-IDF and Word2Vec embeddings, and feature selection using PCA and ANOVA.

3.1 Promise Expanded Dataset

The Promise Expanded Dataset is a collection of software development requirements in various NASA software projects. It comprises 969 instances with clearly defined attributes, including ProjectID, RequirementText, and RequirementType.

The PROMISE Expanded Dataset consists of 969 instances with three key attributes:

i. **ProjectID**: Identifies the project.

ii. **RequirementText**: Textual representation of the requirements.

iii. **RequirementType**: Categorizes the requirements into Functional and Non-Functional types, with specific categories like Performance, Usability, and Security.

To better understand the dataset, we conducted an initial exploration, focusing on the distribution and characteristics of the RequirementType attribute.Figure 1 shows a bar plot visualizing the class distribution in the Promise expanded dataset.

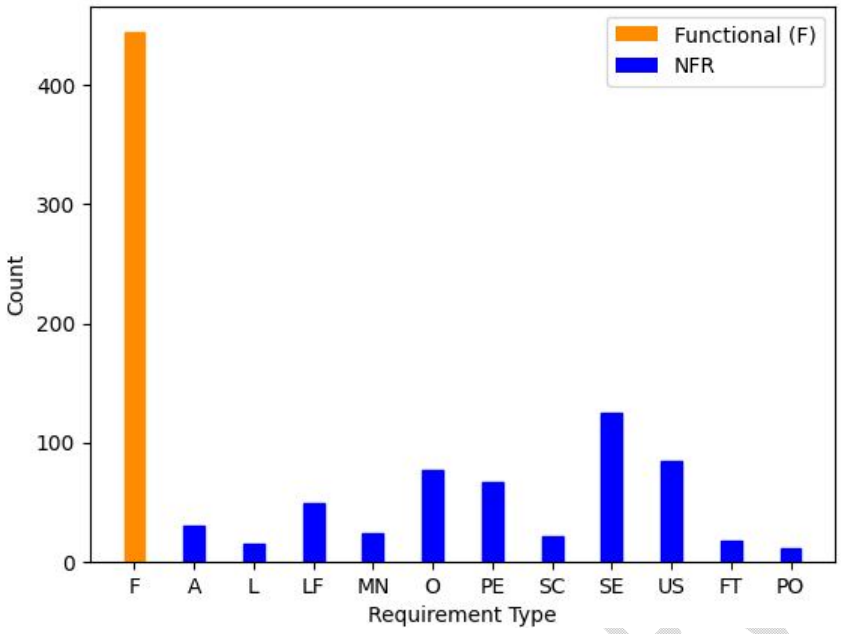

Figure 1: Promise expanded dataset class distribution

The bars in Figure 1 represent the distribution of various requirement types with their respective sample counts. The **Functional Requirements (F)** category has the tallest bar, indicating its significant dominance in the dataset. In contrast, the other bars represent different **Non-Functional Requirement (NFR)** categories, each with much smaller counts compared to the functional category.

This figure demonstrates the dataset's imbalance, with Functional Requirements (F) being overwhelmingly prevalent, while several Non-Functional Requirement (NFR) categories, such as Security (SE) and Usability (US), have relatively more presence compared to others like Fault Tolerance (FT) and Portability (PO).

Acategorization of the dataset into Functional and Non-Functional Requirementsis presented in Table 1.

Table 1: Distribution of Requirement Types in Promise Expanded Dataset

Requirement Type	Count
Functional (F)	444
Availability (A)	31
Legal (L)	15
Look-and-feel (LF)	49
Maintainability (MN)	24
Operability (O)	77
Performance (PE)	67
Scalability (SC)	22
Security (SE)	125

Requirement Type	Count
Usability (US)	85
Fault Tolerance (FT)	18
Portability (PO)	12

This dataset's structured framework and labeled nature makes it suitable for supervised learning approaches.

3.1.1 Balancing the Promise Expanded Dataset with SMOTE

In addressing the class imbalance present in the Promise Expanded Dataset, we employed the Synthetic Minority Over-sampling Technique (SMOTE) to enhance the representation of the minority class. The original dataset comprised 969 samples with 2113 features, where the distribution of classes was significantly skewed. This imbalance posed a challenge for effective model training, as classifiers tend to be biased towards the majority class, leading to suboptimal performance in predicting the minority class.

To mitigate this issue, we implemented SMOTE, which generates synthetic samples for the minority class by interpolating between existing minority class instances. Specifically, for each minority class sample, SMOTE identifies its k-nearest neighbors (in our case, k=5) and creates new synthetic examples along the line segments connecting the minority sample to its neighbors. This approach not only increases the number of minority class samples but also helps in expanding the decision boundary, allowing the classifier to generalize better.

After applying SMOTE, the dataset was resampled to a total of 5328 samples while maintaining the original feature dimensionality of 2113. The resampling process effectively balanced the class distribution, providing a more equitable representation of both classes. The resulting feature matrix is summarized in Table 2, which presents the shape of the original and resampled datasets.

Table 2: Dataset Shapes Before and After SMOTE

Dataset	Number of Samples	Number of Features
Original Dataset	969	2
Resampled Dataset	5328	2

The resampled dataset was then subjected to Principal Component Analysis (PCA) to reduce its dimensionality for visualization and further analysis. The PCA transformation yielded a new shape of (5328, 2), allowing us to visualize the distribution of the samples in a two-dimensional space.

Table 3presents the PCA transformed data, with each sample's index and its corresponding values for the two principal components.

Table 3: Sample of PCA Transformed Data.

Sample Index	Principal Component 1	Principal Component 2
1	0.04535025	0.05334709
2	0.00410043	-0.02551591
3	0.03961297	0.00944405
4	0.23413237	0.07706761
5	0.05658175	0.03019585

Table 4 presents a sample of the PCA transformed data after applying SMOTE to the Promise Expanded Dataset. Each row corresponds to a synthetic sample generated by SMOTE, which has been transformed into a two-dimensional space using Principal Component Analysis (PCA). This transformation reduces the dimensionality of the dataset while preserving variance, facilitating easier visualization and analysis of the data distribution. The two principal components (PC1 and PC2) capture the most significant directions of variance, summarizing the information from the original high-dimensional feature space.

The values in the table illustrate the positioning of synthetic samples created by SMOTE in this reduced feature space. By generating synthetic examples that lie between existing minority class samples, SMOTE fosters a more balanced representation of the minority class. This balance is crucial for improving classifier performance, as it enables them to learn from a richer set of examples, thereby enhancing their ability to generalize and accurately predict outcomes for the minority class.

The results of the PCA transformation are presented in Table 4

Table 4: PCA Transformed Dataset Shape

Transformed Dataset	Number of Samples	Number of Features
PCA Transformed Data	5328	2

The application of SMOTE not only addressed the class imbalance but also facilitated the generation of a more robust training dataset. This enhancement is crucial for improving the sensitivity and specificity of classifiers, particularly in scenarios where the minority class is of significant interest, such as in medical diagnosis or fraud detection.

To further illustrate the effectiveness of SMOTE, we provide a scatter plot of the PCA-transformed data, which visually represents the distribution of the samples across the two principal components. Figure 2 highlights the improved separation between the classes after applying SMOTE.

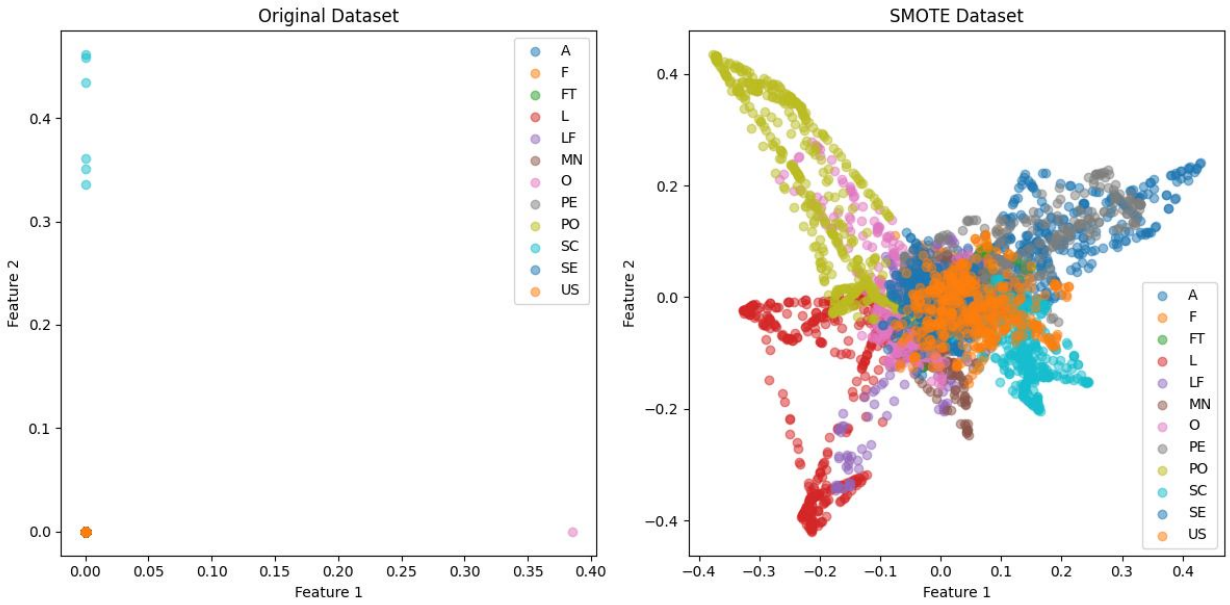

Figure 2: Scatter Plot of PCA-Transformed Data

The integration of SMOTE into the preprocessing pipeline has proven to be a valuable strategy for balancing the Promise Expanded Dataset, thereby enhancing the overall performance of subsequent machine learning models. The results underscore the effectiveness of SMOTE in creating a balanced dataset that supports the development of more accurate predictive models.

3.1.2 Expected Table Result After Applying SMOTE

Applying SMOTE results in a balanced dataset with a more uniform distribution across the requirement types, allowing for more accurate and reliable classification models. The distribution table of the dataset after applying SMOTEis shown in Table 5.

Table 5:Distribution of Requirement Types After Applying SMOTE

Requirement Type	Count
Functional (F)	444
Availability (A)	444
Legal (L)	444
Look-and-feel (LF)	444
Maintainability (MN)	444
Operability (O)	444
Performance (PE)	444
Scalability (SC)	444
Security (SE)	444
Usability (US)	444
Fault Tolerance (FT)	444
Portability (PO)	444

By applying SMOTE, each requirement type is adjusted to have an equal count of 444, aligning the dataset for better performance in training supervised learning models and reducing biases associated with class imbalance.

3.2 Coquina Dataset

The Coquina dataset contains 1438 rows derived from XML-formatted tender documents. It includes attributes such as ProjectID and RequirementText but lacks explicit labels for requirement types. This dataset's diverse project domains and real-world context offer valuable insights into practical requirements. It provides a larger volume of data compared to the PROMISE Expanded dataset. A summary of attributes in Coquina dataset is presented in Table 6.

Table 6: Summary of Attributes in Coquina Dataset

Attribute	Description
ProjectID	Unique identifier for each project.
RequirementText	The textual description of the requirement.

The Coquina dataset's diverse domains (e.g., healthcare, banking, insurance, and web development) and real-world context complement the PROMISE dataset by offering practical examples and a broader range of requirements. The dataset lacks explicit labels for requirement types, presenting an opportunity for classification techniques to infer labels from textual content.

3.3 The Flow Diagram

The flow diagram illustrates the process of feature engineering for managing agile requirements of past projects from the dataset using semantic strategies. The dataset was first tokenized and later balanced using SMOTE. The processed data was then subjected to two main analyses: Topic Feature Extractionand LDA Analysis. The features derived from these analyses are further refined using TF-IDF and Word2Vec for feature extraction, followed by PCA and ANOVA for feature selection. Finally, these selected features are fed into a machine learning model for requirement management. The flow diagram for the study is presented in Figure 3.

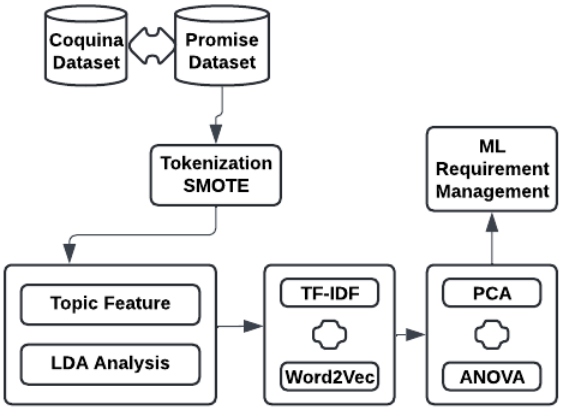

Figure 3: Flow Diagramfor Semantic Analysis Strategies for Agile Requirement Management

3.4. Feature Extraction

During preprocessing, we applied the following steps to the RequirementText data:

i. **Stop Word Removal:** All common stop words were removed to reduce noise and focus on meaningful words.

ii. **Punctuation Removal:** Punctuation marks were stripped from the text to ensure consistency in tokenization.

iii. **Lowercasing:** All text was converted to lowercase to maintain uniformity and avoid case-sensitive discrepancies.

Next, tokenization was performed to split the RequirementText into individual tokens, allowing for more effective feature extraction and analysis.

Tokenization

For both datasets, word-based tokenization is employed. This method breaks down the requirement text into individual words or tokens, facilitating analysis. For example, the requirement "The system shall refresh the display every 60 seconds." is tokenized into ["The", "system", "shall", "refresh", "the", "display", "every", "60", "seconds", "."]. Table 7 shows how each document row in the dataset was tokenized.

Table 7: Example of Tokenization

Original Text	Tokenized Text
"User should be able to log in"	["User", "should", "be", "able", "to", "log", "in"]

Stop Word Removal:This stage may also involve additional preprocessing steps such as stop word removal, where common words like "the" and "and" are filtered out as they do not contribute significant meaning to the analysis. Mathematically, the removal process is presented in Equation 1

$$f(T) = T - S \qquad \text{Equation 1}$$

where T is the vector of tokens and S is the vector of stop words.

Text Cleaning: Includes removal of punctuation and handling hyphens. Punctuation is removed to ensure clean tokens, and hyphens are managed based on their impact on sentiment analysis.Text cleaning is another critical step, which includes removing punctuation, special characters, and numbers that might clutter the dataset. Common words that do not contribute significantly to meaning are removed to reduce noise.

Once tokenization is complete, the resulting tokens are used to build the feature matrix, which will be the basis for feature extraction techniques. This step is crucial for ensuring that the data is in a suitable format for the subsequent stages of analysis, including TFIDF and Word2Vec transformations.

3.5 Latent Dirichlet Allocation (LDA) Analysis on Stop Word Removal

Stop words are common words that carry little meaningful information, such as "is", "the", "and". Removing these words can help in focusing on the most relevant terms during text analysis. This analysis examines the impact of stop word removal on Latent Dirichlet Allocation (LDA) topic modeling, text classification, and cosine similarity using three sample sets from the Promise Expanded dataset.

Latent Dirichlet Allocation (LDA) is a generative probabilistic model used to discover the latent topics within a collection of documents. Our dataset is considered a document with a mixture of topics and each topic is a mixture of words.

We measured the impact by leveraging LDA's ability to identify topics within our dataset, gaining insights into how these topics are distributed across the dataset and examining how they correlate with different types of requirements. This analysis was conducted both before and after significant preprocessing steps, such as stop word removal

The LDA model was trained on the preprocessed text data to extract latent topics. We set the number of topics to 10, with an alpha value of 0.1 to control the distribution of topics across documents, and a beta value of 0.01 to manage the distribution of words across topics.

3.5.1 Dataset Samples

Sample 1:

A. **Original:** "The system shall refresh the display every 60 seconds."
B. **Without Stop Words:** ["system", "shall", "refresh", "display", "every", "60", "seconds"]

Sample 2:

A. **Original:** "The application shall match the color of the schema set forth by Department of Homeland Security."
B. **Without Stop Words:** ["application", "shall", "match", "color", "schema", "set", "forth", "Department", "Homeland", "Security"]

Sample 3:

A. **Original:** "If projected the data must be readable. On a 10x10 projection screen 90% of viewers must be able to read Event / Activity data from a viewing distance of 30."
B. **Without Stop Words:** ["projected", "data", "must", "readable", "10x10", "projection", "screen", "90", "%", "viewers", "able", "read", "Event", "Activity", "viewing", "distance", "30"]

3.5.2 Latent Dirichlet Allocation (LDA) Analysis

Sample 1:

Topic Distribution (Before Removal): Topic 10 ("information") had the highest probability of 0.4857.

Sample 2:

Topic Distribution (Before Removal): Topic 7 ("available") had the highest probability of 0.9182.

Sample 3:

Topic Distribution (Before Removal): Topic 5 ("customer") had the highest probability of 0.9471.

3.6Impact of Stop Word Removal

Table 8 shows the Bag of Words (BoW) matrix for Sample 1 requirement before the removal of stop words, illustrating the frequency of terms used.

Table 8: Bag of Words Matrix for Sample 1

Document	System	shall	Refresh	display	every	60	seconds
Requirement	1	1	1	1	1	1	1

Table 9 presents the topic distribution for Sample 1 requirement before stop word removal, highlighting the top words and their probabilities within each topic

Table 9: Topic Distribution for Sample 1 Requirement Before Removal of Stop Words.

Topic Index	Topic	TopWords	Topic Distribution
0	Topic 1	'user'	0.01666861
1	Topic 2	'shall'	0.01666791
2	Topic 3	'allow'	0.01667039
3	Topic 4	'enable'	0.01666936
4	Topic 5	'users'	0.01666877
5	Topic 6	'select'	0.01666929
6	Topic 7	'report'	0.01667186
7	Topic 8	'send'	0.01666886
8	Topic 9	'enter',	0.38092515
9	Topic 10	'information'	0.48571981

Table 10 displays the Bag of Words (BoW) matrix for Sample 2 requirement before stop word removal, detailing the term frequencies.

Table 10: Bag of Words Matrix for Sample 2

Document	application	shall	Match	color	schema	set	forth	Department	Homeland	Security
Requirement	1	1	1	1	1	1	1	1	1	1

Table 11 provides the topic distribution for Sample 2 requirement before the removal of stop words, showcasing the primary words and their associated probabilities for each topic.

Table 11: Topic Distribution for Sample 2 requirement Before Removal of Stop Words.

Topic Index	Topic	TopWords	Topic Distribution
0	Topic 1	'shall'	0.0090939
1	Topic 2	'user'	0.00909693
2	Topic 3	'users'	0.00909187
..	..	..	..
7	Topic 8	'use'	0.00909238
8	Topic 9:	'display'	0.00909739
9	Topic 10	'disputes'	0.00909338

Table 12 illustrates the topic distribution for Sample 3 requirement prior to stop word removal, focusing on the significant words and their probabilities in each topic.

Table 12: Topic Distribution for Sample 3 requirement Before Removal of Stop Words.

Topic Index	Topic	TopWords	Topic Distribution
0	Topic 1	'shall'	0.00588262
1	Topic 2	, 'product'	0.00588282
2	Topic 3	'user'	0.00588295
3	Topic 4	'able'	0.00588296
4	Topic 5	'customer'	0.94705368
5	Topic 6	'access'	0.00588312
6	Topic 7	'request',	0.00588309
7	Topic 8	'supervisors'	0.00588243
8	Topic 9:	'ensure'	0.00588351
9	Topic 10	'time'	0.00588281

Table 13 demonstrates the impact of stop word removal on topic modeling by comparing the top words and topic changes across several sample requirements.

Table 13: Impact of changes in Topics Modelling

RequirementText	label	kB	kA	TopWords
'The system shall refresh the display every 60 seconds.'	PE	9	9	['user', 'shall', 'allow', 'enable', 'users', 'select', 'report', 'send', 'enter', 'information']
'The application shall match the color of the schema set forth by Department of Homeland Security'	LF	6	4	['shall', 'user', 'users', 'product', 'time', 'allow', 'available', 'use', 'display', 'disputes']
'If projected the data must be readable. On a 10x10 projection screen 90% of viewers must be able to read Event / Activity data from a viewing distance of 30'	US	4	2	['shall', 'product', 'user', 'able', 'customer' 'access', 'request', 'supervisors', 'ensure', 'time']
'The product shall be available during normal business hours. As long as the user has access to the client PC the system will be available 99% of the time during the first six months of operation.'	A	6	3	['shall', 'user', 'users', 'product', 'time', 'allow', 'available', 'use', 'display', 'disputes']
'If projected the data must be understandable. On a 10x10 projection screen 90% of viewers must be able to determine that Events or Activities are occuring in current time from a viewing distance of 100'	US	4	8	['shall', 'product', 'user', 'able', 'customer' 'access', 'request', 'supervisors', 'ensure', 'time']
'The product shall ensure that it can only be accessed by authorized users. The product w be able to distinguish between authorized and unauthorized users in all access attempts'	SE	3	7	['shall', 'product', 'player', 'shot', 'defensive', 'offensive', 'grid', 'ship', 'hit', '99']
'The product shall be intuitive and self-explanatory. 90% of new users shall be able t start the display of Events or Activities within 90 minutes of using the product.'	US	6	4	['shall', 'user', 'users', 'product', 'time', 'allow', 'available', 'use', 'display', 'disputes']

Removing stop words led to a more diverse topic distribution in the LDA analysis. The results show a clearer representation of underlying topics, as seen in the increased diversity of topics after stop word removal, improving topic modeling accuracy.

4.0Results

The results of feature extraction using the different techniques are presented in this Section.

4.1 Feature Extraction

Feature extraction is a critical step in transforming textual data into a numerical format that can be effectively used by machine learning models. In this stage, two prominent techniques are employed: TFIDF (Term Frequency-Inverse Document Frequency) and Word2Vec embeddings. This hybrid method provides a nuanced representation by integrating both term frequency and semantic relationships.

4.1 TFIDF (Term Frequency-Inverse Document Frequency)

The TFIDF score for a term t in a document d within a corpus D is calculated as follows;

Term Frequency (TF): Measures how frequently a term appears in a document. The formula for TF is presented in Equation 2.

$$TF(t,d) = \frac{Total\ number\ of\ terms\ t\ in\ document\ d}{Frequency\ of\ term\ t\ in\ document\ d} \quad \text{Equation 2}$$

i. **Inverse Document Frequency (IDF)**: Measures how important a term is across the entire dataset. The formula for IDF is presented in Equation 3.

The inverse of the document frequency is mathematically represented as follows:

$$idf_i = log \frac{total\_requirements}{total\_requirements\_with\_termitotal\_requirements} \quad \text{Equation 3}$$

Combining these two metrics, the TF-IDF feature vector can be defined in Equation 4:

$$TF - IDF(term_{i,j}) = tf_{i,j} \times idf_i \quad \text{Equation 4}$$

Here, *tf* represents the frequency of the term in the document, and *idf* is the inverse of the document frequency for term i and document j.Therefore, the term frequency-inverse document frequency (TF-IDF) is given in Equation5

$$tf - idf(t_d, i_j) = \frac{tf(t_d,i_j) log\ (\frac{|D|}{df(t_i)})}{d_j} \quad \text{Equation 5}$$

The algorithm to compute TF-IDF is given below:

Step 1: Inverse Document Frequency (IDF) Calculation

Count the number of times each word appears in a sentence.
Divide by the total number of words in the sentence.

Step 2: Inverse Document Frequency (IDF) Calculation

Count the number of sentences each word appears in.
Divide the total number of sentences by this number.
Take the logarithm of the result.

Step 3: Weighted TF-IDF Calculation
Multiply the TF and IDF values for each word.

To compute the weighted TF-IDF Matrix, we would have to compute each TF-IDF across their i, j word. The word **system** in the first sentence of our first five requirement in the Promise explained corpus given below,

"system shall refresh display every 60 seconds",

"application shall match color schema set forth department homeland security",

"if projected data must readable on 10x10 projection screen 90 % viewers must able read event / activity data viewing distance 30",

"the product shall available normal business hours as long user access client pc system available 99 % time first six months operation",

"if projected data must understandable on 10x10 projection screen 90 % viewers must able determine events activities occurring current time viewing distance 100"

$$TF_{system,1} = \frac{Number\ of\ times\ 'system'\ apears\ in\ requirement\ 1}{Total\ words\ in\ requirement\ 1} = \frac{1}{6} \quad \text{Equation 6}$$

$$IDF_{system} = \log\frac{Total\ number\ of\ requirement}{Number\ of\ requirement\ with\ 'system'} = \log\frac{5}{2} \quad \text{Equation7}$$

$$TF - IDF(term_{system,1}) = TF_{system,1} \times IDF_{system} = \left(\frac{1}{6}\right) \times \log\left(\frac{5}{2}\right) \quad \text{Equation 8}$$

$$TF - IDF(term_{system,1}) = \frac{\log(2.5)}{6} = \approx \frac{0.39794}{6}\frac{1}{6} \times 0.39794 \approx 0.06632 \quad \text{Equation 9}$$

The output of the TF-IDF stage is a weighted matrix for the Promise expanded dataset, with dimensions 969 x 2060. The matrix is shown below:

$$Requirements\ (969 rows) \begin{matrix} \text{Word 1} & \text{Word 2}\ldots.. & & & & \text{Word 2060} \end{matrix} \begin{pmatrix} TF-IDF_{1,1} & TF-IDF_{1,2} & .. & .. & .. & TF-IDF_{1,2060} \\ TF-IDF_{2,1} & TF-IDF_{2,2} & .. & .. & .. & TF-IDF_{2,2060} \\ .. & .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. & .. \\ TF-IDF_{969,1} & TF-IDF_{969,2} & .. & .. & .. & TF-IDF_{969,2060} \end{pmatrix}$$

The weighted TF-IDF matrix is sparse, with many zero entries due to the large number of unique terms and limited presence in individual documents. To simplify, only the top 10 words by their total TF-IDF score are presented. Figure 4 displays a word cloud illustrating the importance of these terms across the dataset.

Figure 4: Wordcloud of Words across Requirements

The result of the top N=10 weighted T-IDF is presented in Table 14.

Table 14: Weighted TF-IDF of Top N=10 words

shall	The	system	product	user	Allow	users	must	able	information
0.1155	0.1171	0.1513	0	0	0	0	0	0	0
0.0763	0.0774	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0.2405	0.1229	0
0.0614	0.0622	0.0804	0.1065	0.1143	0	0	0	0	0
0	0	0	0	0	0	0	0.2356	0.1203	0
0.0709	0.0720	0	0.2462	0	0	0.3295	0	0.1578	0
0.1328	0.0673	0	0.2304	0	0	0.1542	0	0.1477	0
0.0855	0.0868	0	0.1484	0	0	0	0	0	0
0.0855	0.0867	0.1120	0	0	0	0	0	0	0
0.1154	0.1170	0.1512	0	0	0	0	0	0	0
0.1372	0.1391	0.1797	0	0	0	0	0	0	0
0.1481	0.1502	0.1940	0	0	0	0	0	0	0
0.0701	0.0711	0	0	0	0	0	0	0	0
0	0.0740	0	0	0	0	0	0	0	0

4.2 Word2Vec

Word2Vec was employed to generate vector representations of requirements text, enabling the capture of semantic relationships between words. We compared two models: Continuous Bag of Words (CBoW) and SkipGram, ultimately selecting the SkipGram model for its superior performance in identifying semantic similarities between requirements. This decision was based on its higher similarity scores, which better distinguished requirements within and across different classes, as demonstrated in Table 15. The effectiveness of the SkipGram model underscores its ability to enhance the accuracy of requirement classification in our adaptive management system.

Table 15: Word2Vec Measurement of Semantic Similarity

Requirement 1	Requirement 2	Class	CBoW	SkipGram
['"The", 'look', 'and', 'feel', 'of',	['"The", 'product', 'shall', 'have', 'a',	LF	0.9984	0.9995

Requirement 1	Requirement 2	Class	CBoW	SkipGram
'the', 'system', 'shall', 'conform', 'to', 'the', 'user', 'interface', 'standards', 'of', 'the', 'smart', 'device', '.']	'consistent', 'color', 'scheme', 'and', 'fonts', '.']			
['"The", 'system', 'shall', 'provide', 'a', 'history', 'report', 'of', 'changes', 'made', 'to', 'the', 'Activity', 'or', 'Event', 'data']	['"The", 'system', 'shall', 'display', 'both', 'the', 'active', 'and', 'completed', 'order', 'history', 'in', 'the', 'customer', 'profile', '.']	PE	0.9979	0.9996
['"The", 'search', 'results', 'shall', 'be', 'returned', 'no', 'later', '30', 'seconds', 'after', 'the', 'user', 'has', 'entered', 'the', 'search', 'criteria']	['"The", 'search', 'for', 'the', 'preferred', 'repair', 'facility', 'shall', 'take', 'no', 'longer', 'than', '8', 'seconds', '.', 'The', 'preferred', 'repair', 'facility', 'is', 'returned', 'within', '8', 'seconds']	PE	0.9983	0.9995

Table 15 presents the measure of similarity between eachpair of the requirement. Figure 5 is a scatter plot that compares semantic similarity scores between pairs of requirements using two models: CBoW and SkipGram. Each point shows the similarity score from CBoW on the x-axis and the score from SkipGram on the y-axis.

Red Points (LF - Look-and-Feel): For example, the pair R1-R2 has high similarity scores of 0.9984 (CBoW) and 0.9995 (SkipGram), indicating strong agreement between the models.

Blue Points (PE - Performance): For instance, the pair R3-R4 has scores of 0.9979 (CBoW) and 0.9996 (SkipGram), also showing high similarity but slightly less agreement compared to the LF category.

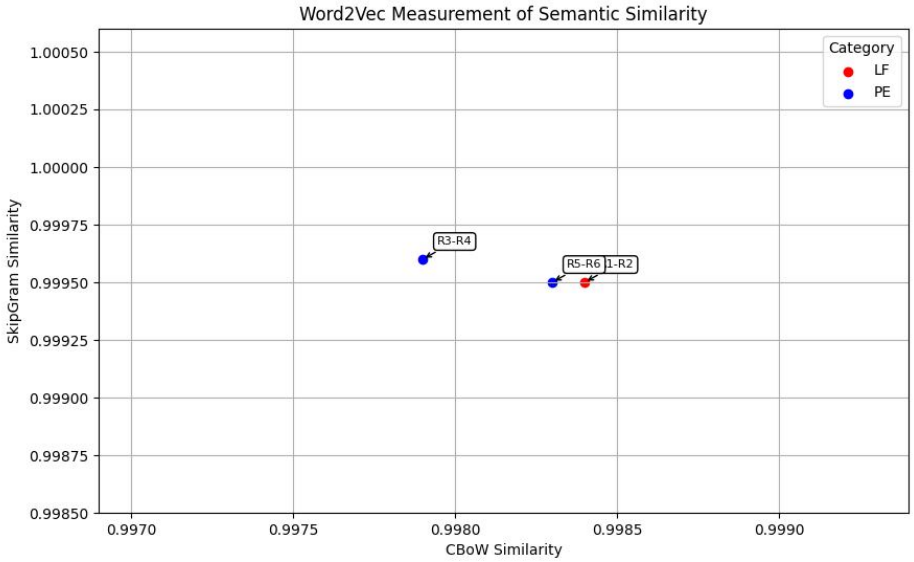

Figure 5: Word2vec Measurement of Semantic Similarity

The vector representation for word w is exemplified in Table 16.

Table 16: Example of Word2Vec Vectors:

Word	Vector Representation
"login"	[0.12, 0.34, -0.45, ...]
"user"	[0.22, -0.14, 0.33, ...]

Word2Vec provides two main models: Continuous Bag of Words (CBOW) and Skip-Gram.

i. **CBOW Model:** Predicts the target word based on its surrounding context words. The objective is to maximize the probability of the target word given its context:

$$P(w_t \mid w_{t-n}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+n})$$

ii. **Skip-Gram Model:** Predicts the surrounding context words given a target word. The goal is to maximize the probability of the context words given the target word:

$$P(w_{t-n}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+n} \mid w_t)$$

In Word2Vec, each word is represented as a dense vector in a high-dimensional space, where semantically similar words have similar vector representations. The training parameters typically include vector size, window size, and minimum count, which control the dimensionality of the vectors, the size of the context window, and the minimum frequency of words to be considered, respectively.

4.3 TF-IDF Weighted Word2Vec

In Sections 3.5.1 and 3.5.2, we discussed TF-IDF and Word2Vec for converting text into numerical data. TF-IDF highlights word importance but lacks semantic context, while Word2Vec captures semantic meaning but ignores term importance. To combine their strengths, we propose a TF-IDF weighted Word2Vec approach. This method integrates TF-IDF's term importance with Word2Vec's semantic understanding by weighting Word2Vec vectors with TF-IDF scores, resulting in a more contextually relevant representation of each document. The final TF-IDF weighted Word2Vec matrix has dimensions 969 x 100, with 969 documents and 2060 unique words. The TF-IDF weighted Word2Vec representation is calculated through matrix multiplicationof the TF-IDF representation and Word2Vec representations as shown in Equation 10

$$R = T \cdot W \qquad \text{Equation 10}$$

Each row in *R* is computed in Equation 3.19

$$\mathrm{R}_i = \sum_{j=1}^{2060} T_{(word,i)} \cdot W_{word} \qquad \text{Equation 11}$$

Where:

R_i is the *i*-th row in *R*

$T_{(word,i)}$ is the TF-IDF weight of the i-th document

W_{word} is the Word2Vec vector of the word

The TF-IDF weighted Word2Vec representation matrix R of size 969 x 100 is computed for each document. For example, the first requirement ("system shall refresh display every 60 seconds") is represented by expanding the equation as:

$$wR(d_1) = T_{system,1}W(system) + T_{shall,1}W(shall) + T_{refresh,1}W(refresh) + T_{display,1}W(display) + T_{every,1}W(every) + T_{60,1}W(60) + T_{seconds,1}W(seconds)$$

This process is repeated for the entire dataset, resulting in a final matrix with 969 rows and 100 features.

5.0 Feature Selection

In the feature selection process, PCA and ANOVA were used to refine the TF-IDF weighted Word2Vec data obtained in Section 4.3.

5.1 PCA (Principal Component Analysis)

Principal Component Analysis (PCA) was employed to reduce the high-dimensional data derived from the TF-IDF weighted Word2Vec representations, resulting in a more efficient and interpretable dataset for further analysis and modeling.

i. **Dimensionality Reduction:** The initial dataset consisted of a high-dimensional feature space with 100 features for each of the 969 requirements, generated from the TF-IDF weighted Word2Vec embeddings. PCA was applied to reduce these 100 features to 30 dimensions, balancing the retention of significant variance in the data with the need for computational efficiency.

ii. **Feature Extraction:** Through PCA, new features (principal components) were extracted, representing linear combinations of the original features. These principal components were uncorrelated and captured the most variance within the dataset, highlighting the most informative aspects of the data. This transformation set the stage for enhancing the performance of subsequent machine learning models.

5.2 ANOVA (Analysis of Variance)

Following PCA, Analysis of Variance (ANOVA) was applied to evaluate the relevance of the transformed features in relation to the class labels (target variable):

- **Significance Testing:** Features were assessed based on their F-statistic values, with a significance threshold set at p-value < 0.05. High F-statistic values indicate features that are most relevant to the classification task.

- **Feature Selection:** Based on the F-statistic results, significant features were selected for further analysis and model training, enhancing the predictive performance and reducing computational complexity.

The Mathematical Representation of the ANOVA F-statistic for feature f is given in Equation 12.

$$F = \frac{Within-Class\ Variance}{Between-Class\ Variance} \quad \text{Equation 12}$$

F-Statistic Results: The F-statistic results for the features selected from the TF-IDF weighted Word2Vec embeddings are presented in Table 17. This table includes the Sum of Squares Between (SSB), Sum of Squares Within (SSW), Mean Square Between (MSB), Mean Square Within (MSW), and the F-statistic for each feature.

Table 17: F-Statistic Results for Selected Features

Features	SSB	SSW	MSB	MSW	F-Statistic
Feature2	1.89e-05	3.47e-05	6.29e-06	2.89e-06	2.1732
Feature3	0.001066	0.00022	0.00036	1.79e-05	19.8414
Feature9	0.00091	3.23e-05	0.00030	2.69e-06	112.6935
Feature14	6.55e-05	2.134e-05	1.55170	1.78e-06	8.7262

Features from the PCA results are ranked based on their F-statistic values. A higher F-statistic indicates a more significant difference between groups, suggesting that the feature is more relevant.A significance threshold set at a value of p-value < 0.05is applied to select features with statistically significant differences.

Selection of Significant Features: Based on the F-statistic results, features with higher F-statistic scores indicate a greater ability to distinguish between classes. In this case, **Feature9** and **Feature3** were identified as the most significant features due to their high F-statistic values (112.6935 and 19.8414, respectively). These features were selected for further analysis and model training, while **Feature2** and **Feature14** were deemed less significant and not selected.

Top Selected Features: The top ten features selected from the TF-IDF weighted Word2Vec embeddings across different datasets are summarized in Table 18. TheTable lists the features that were most relevant to the class labels.

Table 18: Top Ten Features

Dataset	Count	Top Ten Features
tfidf_w2v-12	100	feature53, feature62, feature92, feature76, feature27, feature25, feature6, feature44, feature52, feature77
tfidf_w2v-11	100	feature53, feature27, feature6, feature62, feature44, feature9, feature57, feature52, feature25, feature89
tfidf_w2v-2	11	feature25, feature92, feature2, feature76, feature14, feature66, feature67, feature53, feature36, feature3

Table19 summarizes the ANOVA results, indicating the significance of each feature based on the calculated p-values. Features with lower p-values < 0.05are considered statistically significant in distinguishing between classes.

Table 19: ANOVA Results

Feature	F-Statistic	p-value
Feature2	2.1732	0.1442
Feature3	19.8414	0.0000605
Feature9	112.6935	0.00000000469
Feature14	8.7262	0.0024

6.0 Conclusion

This study demonstrates the effectiveness of integrating semantic feature extraction, advanced feature selection, and topic modeling to enhance adaptive management strategies in agile software development. By employing TF-IDF weighted Word2Vec for feature extraction and Latent Dirichlet Allocation (LDA) for topic modeling, we significantly improved the semantic representation of requirements and the clarity of the requirement text used to describe user needs. To address the challenge of high-dimensional embeddings from TF-IDF weighted Word2Vec, Principal Component Analysis (PCA) was applied to reduce the feature space from 100 dimensions to 30. Additionally, ANOVA was used to select the top 3 features that best capture the semantic characteristics of the dataset.

Practical Implications: The methodologies applied in this study have practical implications for the software development industry, particularly in Agile environments. The improved precision and recall scores for functional requirements (F) highlight the ability to manage the predominant class effectively. Additionally, SMOTE's application helps address class imbalance, improving the handling of minority requirement types. These approaches can be directly applied in real-world scenarios to enhance the accuracy and efficiency of requirement management, contributing to more reliable and adaptive software development processes.

Limitations: Despite the progress made, the study has certain limitations. It focused primarily on text-based requirements, potentially overlooking the complexity of non-textual data such as diagrams or models in Agile projects. Moreover, while SMOTE was effective in balancing the dataset, the synthetic generation of minority class examples may not fully capture the nuances of actual requirements.

Future Work: Future research should explore integrating multi-modal data, including visual and diagrammatic representations, to extend the model's applicability. Investigating alternative techniques for handling class imbalance beyond SMOTE could lead to more robust models. Additionally, the features selected in this study will be used for predicting requirements via machine learning models, enhancing adaptive requirement management strategies. Expanding the study to include a broader range of datasets from various domains could also provide insights into the generalizability of the proposed methods. Addressing these gaps will advance the development of adaptive requirement management strategies in agile environments.

Disclaimer (Artificial intelligence)

Option 1:

Author(s) hereby declare that NO generative AI technologies such as Large Language Models (ChatGPT, COPILOT, etc) and text-to-image generators have been used during writing or editing of manuscripts.

Option 2:

Author(s) hereby declare that generative AI technologies such as Large Language Models, etc have been used during writing or editing of manuscripts. This explanation will include the name, version, model, and source of the generative AI technology and as well as all input prompts provided to the generative AI technology

Details of the AI usage are given below:

1.

2.

3.

References

Abbas, M., Ferrari, A., Shatnawi, A., Enoiu, E. P., & Saadatmand, M.(2021). Is requirements similarity a good proxy for softwaresimilarity? An empirical investigation in industry. In RequirementsEngineering: Foundation for Software Quality (pp. 22-37). SpringerInternational Publishing.

Asad, K., & Muqeem, M. (2022). Critical analysis of requirementmanagement in agile development. Empirical Software Engineering,26(28).

Bing, L., & Xiuwen, N. (2022). NFRNet: A deep neural network forautomatic classification of non-functional requirements. InProceedings of the 2022 IEEE 29th International RequirementsEngineering Conference (RE).

Canedo, E., & Mendes, B. (2020). Software requirements classificationusing machine learning algorithms. Entropy, 22(9), 1057.

Franch, X., Seyff, N., Oriol, M., Fricker, S., Groher, I., Vierhauser,M., & Wimmer, M. (2020). Towards integrating data-driven requirementsengineering into the software development process: A vision paper. In26th International Working Conference, Italy (pp. 135-144).

Kurtanović, Z., & Maalej, W. (2017). Automatically classifyingfunctional and non-functional requirements using supervised machinelearning. In 2017 IEEE 25th International Requirements EngineeringConference (RE).

Navarro-Almanza, R., Juárez-Ramírez, R., & Licea, G. (2017). Towardssupporting software engineering using deep learning: A case ofsoftware requirements classification. In 2017 5th IEEE InternationalConference in Software Engineering Research and Innovation (CONISOFT).

Yang, B., Ma, X., Wang, C., et al. (2023). User story clustering inagile development: A framework and an empirical study. Frontiers ofComputer Science, 17, 176213.