An NLP-Enabled Approach to Semantic Grouping for Improved Requirements Modularity and Traceability

Rahat Izhar[1], Shahid Nazir Bhatti[2], Sultan A. Alharthi[3]
Faculty of Engineering, Chiang Mai University, Chiang Mai, Thailand[1]
Department of Software Engineering, College of Computer Science and Engineering, University of Jeddah, Jeddah 21493, Saudi Arabia[2, 3]

***Abstract*—The escalating complexity of modern software systems has rendered the management of requirements increasingly arduous, often plagued by redundancy, inconsistency, and inefficiency. Traditional manual methods prove inadequate for addressing the intricacies of dynamic, large-scale datasets. In response, this research introduces SQUIRE (Semantic Quick Requirements Engineering), a cutting-edge automated framework leveraging advanced Natural Language Processing (NLP) techniques, specifically Sentence-BERT (SBERT) embeddings and hierarchical clustering, to semantically organize requirements into coherent functional clusters. SQUIRE is meticulously designed to enhance modularity, mitigate redundancy, and strengthen traceability within requirements engineering processes. Its efficacy is rigorously validated using real-world datasets from diverse domains, including attendance management, e-commerce systems, and school operations. Empirical evaluations reveal that SQUIRE outperforms conventional clustering methods, demonstrating superior intra-cluster cohesion and inter-cluster separation, while significantly reducing manual intervention. This research establishes SQUIRE as a scalable and domain-agnostic solution, effectively addressing the evolving complexities of contemporary software development. By streamlining requirements management and enabling software teams to focus on strategic initiatives, SQUIRE advances the state of NLP-driven methodologies in Requirements Engineering, offering a robust foundation for future innovations.**

Keywords*—Requirements Engineering (RE); semantic clustering; sentence-BERT; natural language processing (NLP)

I. Introduction

Requirements Engineering is a cornerstone of software development, focusing on the identification, documentation, and management of requirements that guide the design and implementation of software systems [1]. It ensures alignment between user needs and project goals, providing a foundation for system functionality and quality. However, as software systems grow in complexity and scale, managing requirements effectively becomes increasingly challenging. Issues such as redundancy, inconsistencies, and overlapping functionalities not only complicate design processes but also disrupt modularity, traceability, and efficient project execution [2], [3]. These challenges highlight the need for innovative solutions to streamline requirements management, particularly in large-scale projects.

Conventional approaches to managing requirements rely heavily on manual processes, which are often time-consuming, prone to human error, and inadequate for handling large datasets [4]. These traditional methods, while useful for small-scale projects, fail to meet the demands of modern, dynamic software development, where requirements are frequently updated and involve intricate relationships. Automated techniques, particularly those leveraging Natural Language Processing (NLP), have emerged as promising solutions for addressing these limitations. By analyzing textual requirements for semantic relationships, NLP-based methods can uncover patterns and organize requirements efficiently [5]. However, existing methods face limitations in capturing the subtle semantic relationships within diverse or domain-specific datasets, restricting their ability to handle the complexity of real-world requirements engineering tasks.

To address these gaps, this paper presents SQUIRE (Semantic QUIck Requirements Engineering), a novel and structured methodology aimed at automating the grouping of semantically similar requirements into functional clusters. SQUIRE combines state-of-the-art NLP techniques, such as Sentence-BERT (SBERT) embeddings, hierarchical clustering, and a comprehensive preprocessing pipeline, to analyze and group requirements based on their semantic similarity [4], [6], [12]. By focusing on reducing redundancy, enhancing modularity, and improving traceability, SQUIRE provides a practical solution to key challenges in RE, enabling more efficient and effective system design [7].

The methodology represents a significant advancement in the application of NLP to RE, building on recent trends in natural language understanding and clustering algorithms [9], [10], [17], [18]. The authors' work reflects a broader effort in the research community to leverage the capabilities of NLP for addressing critical challenges in RE, including the need for scalability, semantic analysis, and automation [19]. Recent developments in NLP, particularly transformer-based models such as Sentence-BERT (SBERT), have enabled significant improvements in the ability to capture the semantic meaning of textual data, making them well-suited for addressing RE challenges.

The effectiveness of SQUIRE is evaluated across diverse, real-world datasets representing distinct functional domains, including attendance management, e-commerce, and school operations. The methodology's performance is assessed using quantitative metrics such as cohesion, separation, silhouette scores, and the Davies-Bouldin Index [20], [24], [25]. Visualizations using Principal Component Analysis (PCA) further demonstrate the ability of SQUIRE to organize requirements into meaningful clusters, providing clear insights

into the relationships between requirements. These evaluations highlight the robustness of the proposed approach, showcasing its potential to improve modularity and traceability in varied software development contexts.

This research contributes to the ongoing integration of NLP into requirements engineering by demonstrating how advanced language models and clustering techniques can address long-standing challenges in the field [5]. By automating the grouping of requirements, SQUIRE reduces the manual effort required in RE, allowing software teams to focus on higher-value activities such as innovation and strategic planning. Furthermore, the domain-agnostic nature of SQUIRE makes it adaptable to various industries, offering a scalable solution for modern software development needs.

SQUIRE advances the state of requirements engineering by introducing a systematic and scalable methodology for semantic clustering. By leveraging the latest advancements in NLP, it bridges the gap between textual complexity and actionable insights, enabling software teams to design more organized, traceable, and modular systems [22]. This work lays the groundwork for further innovations in automated RE, positioning NLP as a central tool in the evolution of software development practices.

The remainder of this paper is organized as follows: Section II delineates a meticulous review of pertinent literature, elucidating seminal advancements and inherent constraints within NLP-driven requirements engineering. Section III expounds upon the proposed SQUIRE methodology, systematically detailing its preprocessing pipeline, embedding generation mechanisms, and clustering paradigms. Section IV articulates the experimental framework, encompassing the evaluation metrics and empirical assessments employed to ascertain the model's efficacy. Section V presents a rigorous discourse on the obtained findings, critically analyzing the methodological robustness and potential limitations. Lastly, Section VI encapsulates the study's principal contributions and delineates prospective avenues for future research.

II. Related Work

Modern software development has made RE more challenging due to the growing complexity of systems and the volume of requirements. Traditional approaches often rely on manual analysis, which can lead to errors and inefficiencies. To overcome these challenges, researchers have introduced techniques like NLP and clustering algorithms. These methods aim to simplify requirement management by automating processes such as grouping and prioritization. This section reviews recent work in applying these techniques to RE, focusing on their impact, limitations, and future potential.

Radwan et al. [8] proposed the CMHR (Conceptual Mapping for Healthcare Requirements) approach to enhance the analysis of non-functional requirements in healthcare systems. The methodology involves clustering requirements using the K-means++ algorithm based on attributes such as priority and suitability, enabling structured visualization through conceptual mapping. Applied to ventilator requirements, the approach achieved a silhouette score of 0.71 and accurately classified new requirements using a Naïve Bayes classifier. However, the study is limited by its small dataset and focus on a single domain, necessitating further validation across other healthcare systems for broader applicability.

Salman et al. [9] explored semantic clustering of functional requirements (FRs) using Agglomerative Hierarchical Clustering (AHC). The study addresses a gap in software requirements engineering, where functional requirements are typically analyzed manually. Existing works focus primarily on non-functional requirements classification, leaving FR clustering largely unexplored. Previous research has used techniques like Support Vector Machines and ontology-based methods for requirement classification, but functional requirements have lacked similar advancements. This paper introduces a semantic similarity-based approach to group FRs, validated across four software projects. While the proposed method achieved promising results, it relies heavily on vocabulary consistency, limiting its applicability across diverse datasets.

Del Sagrado et al. [10] integrate clustering techniques with the MoSCoW method to automate requirements prioritization in software projects. Their methodology, validated on datasets of varying scales (20, 50, and 100 requirements), demonstrates clustering's efficacy in identifying core requirements. While enhancing decision-making, the approach is constrained by dependency considerations and reliance on subjective estimations, leading to variability in algorithmic performance.

Bakar et al. [11] employ Latent Semantic Analysis (LSA) alongside K-Means and Hierarchical Agglomerative Clustering (HAC) to facilitate software requirements reuse in Software Product Lines (SPL). Evaluations on 27 product review documents reveal HAC's superiority in cluster compactness, whereas K-Means marginally outperforms in external validation. However, HAC is preferred for its grouping precision. Sensitivity to domain-specific data and input parameters remains a limitation, with future work directed at refining clustering optimization.

Das et al. [12] introduce PUBER and FiBER, domain-specific sentence embedding models enhancing similarity detection in natural language requirements. Built on the BERT architecture, PUBER trains on the PURE dataset, while FiBER refines it via fine-tuning. Using cosine similarity, FiBER achieves 88.35% accuracy, surpassing Universal Sentence Encoder and RoBERTa by up to 10%. These models advance classification, similarity detection, and reusability, addressing NLP challenges in software engineering.

The study by Elhassan et al. [13] developed an automated conflict detection model using the Mean Shift clustering algorithm to identify and classify requirement conflicts during elicitation. The methodology involved data transformation based on McCall's quality model and clustering requirements into three categories: conflict-free, partial conflict, and conflicted requirements. Results demonstrated accurate clustering, with conflict-free requirements achieving low standard error (SE) values, validating the model's efficiency. Limitations include a small dataset size (207 observations), restricting generalizability, and challenges in data collection from diverse sources. Future work suggests expanding datasets,

applying the model to varied IS environments, and exploring decision tree algorithms for enhanced detection.

The reviewed studies show that using NLP and clustering in Requirements Engineering has great potential to solve key challenges like redundancy and lack of scalability. However, issues such as handling diverse requirements and adapting to different domains remain. These findings suggest the need for smarter, more flexible methods to address these gaps. By building on existing work, approaches like SQUIRE offer a step forward, combining advanced tools with practical solutions for improving how requirements are managed in real-world software projects.

III. Research Methodology

The proposed methodology, SQUIRE offers an automated and efficient framework for grouping semantically similar requirements into distinct functional clusters. By leveraging NLP techniques, embedding models, and hierarchical clustering, SQUIRE addresses key challenges in requirements engineering, such as redundancy reduction, traceability enhancement, and modular system design.

The methodology comprises five structured stages: Preprocessing, Semantic Embedding Generation, Clustering, Evaluation, and Visualization.

A. Data Preprocessing

The preprocessing stage standardizes and simplifies textual requirements, ensuring consistency and reducing noise to prepare data for embedding generation [27]. This involves a series of transformations applied sequentially: converting text to lowercase to eliminate inconsistencies caused by capitalization, removing punctuation marks to simplify content, and filtering out stopwords such as “the” and “shall” that do not add semantic value [30]. The text is then tokenized into individual words or phrases, enabling detailed analysis, and normalized to unify synonyms or domain-specific terms (e.g., “log in” and “sign in” standardized to “log in”). Table I demonstrates the progressive refinement of requirements through these steps.

TABLE I Example Results of Data Preprocessing Pipeline

Original Requirement	Lowercased	Punctuation Removed	Stopwords Removed	Tokenized
"The system shall allow users to log in."	"the system shall allow users to log in."	"the system shall allow users to log in"	"system allow users log in"	["system", "allow", "users", "log", "in"]
"Users can register themselves for access."	"users can register themselves for access."	"users can register themselves for access"	"users register access"	["users", "register", "access"]

The output of preprocessing is a clean, tokenized, and normalized version of each requirement, ready for embedding generation. This ensures that noise is minimized while retaining the semantic essence of the text.

B. Semantic Embedding Generation

To effectively cluster requirements, each preprocessed input is transformed into a dense vector representation using SBERT, a pre-trained transformer-based model optimized for capturing semantic similarity [23]. SBERT excels at encoding contextual information and relationships between words, making it an ideal choice for analyzing and grouping requirements. Specifically, the all-MiniLM-L6-v2 model is utilized, which generates embeddings with a dimensionality of 384 [31]. These embeddings represent the semantic meaning of textual inputs in a high-dimensional vector space. For instance, a requirement like "system allow users log in" is converted into a numerical vector (e.g., E = [0.23, -0.45, 0.67, …, -0.11]). The proximity of two embeddings in this space directly reflects the semantic similarity between their corresponding requirements.

C. Clustering Requirements in Groups

Using the embeddings generated in the previous step, requirements are grouped into clusters through Agglomerative Clustering, a hierarchical method that iteratively merges data points based on similarity [15]. This process relies on Euclidean Distance as the similarity metric, where smaller distances indicate higher semantic similarity between requirements [16], [29]. To ensure optimal clustering, Ward Linkage is employed, which minimizes intra-cluster variance by selecting merges that reduce the overall variance [28]. The variance adjustment is calculated as:

$$\Delta Variance = Variance\ of\ Cluster\ A + Variance\ of\ Cluster\ B - Variance\ of\ Combined\ Cluster$$

For better interpretability and functional modularity, the number of clusters (k) is predetermined based on the dataset’s size and complexity (e.g., k = 4 for smaller datasets). Each resulting cluster represents a distinct functional module, grouping semantically similar requirements while separating unrelated ones [18], [19]. For instance, requirements like *"System allow users log in"* and *"Users register access"* are grouped into Cluster 1, while others such as *"System calculate attendance"* and *"Users generate reports"* fall into separate clusters, reflecting unique functional distinctions. Examples of these clusters is shown in Table II.

TABLE II Clustering Output Example

Requirement	Cluster
Ensure image sliders work properly and link to the restaurant homepage.	1
Match UI with the Android version: icons, formats, sizing, and alignment.	1
Show current prices for discounted items and notify users about deletions.	2
Show discount value on item card and final total price box for % discount items.	2
Connect QR code feature and create deep links for reading the QR code.	3
Ensure every QR code generator has a deep link and forwards to the app store/play store if not installed.	3

D. Evaluation

The quality of clustering results is evaluated using several key metrics to ensure both intra-cluster cohesion and inter-cluster separation. Cohesion measures the semantic similarity of requirements within a cluster, ensuring that grouped requirements share strong relationships [26]. In contrast,

Separation evaluates the dissimilarity between clusters, ensuring functionally distinct requirements are placed in separate groups [20]. The overall clustering quality is assessed using the Silhouette Score, which combines both cohesion and separation into a single metric. Additionally, the Davies-Bouldin Index is used to measure intra-cluster compactness and inter-cluster separation, where lower values indicate better clustering performance [21]. Together, these metrics provide a comprehensive evaluation of clustering effectiveness.

E. Visualization

For interpretability, high-dimensional embeddings are reduced to two dimensions using Principal Component Analysis (PCA). This dimensionality reduction enables the generation of scatter plots where each point represents a requirement, and clusters are distinguished by color [14]. These visualizations provide insights into the clustering structure and relationships among requirements. The SQUIRE methodology integrates preprocessing, semantic embedding generation, clustering, evaluation, and visualization into a cohesive framework for automating requirements analysis. By leveraging SBERT embeddings and hierarchical clustering with a fixed number of clusters, it ensures accurate grouping of semantically similar requirements while enhancing modular design and traceability.

F. Algorithm: SQUIRE (Semantic QUIck Requirements Engineering)

Input:

- $R = \{R_1, R_2, \dots, R_n\}$: A set of n textual requirements.

Output:

- $C = \{C_1, C_2, \dots, C_k\}$: k clusters of semantically similar requirements.

Steps:

1. Preprocessing

Transform each requirement R_i into a standardized form:

- Convert to lowercase: R_i'= lower(R_i).
- Remove punctuation: R_i''= remove_punctuation(R_i').
- Tokenize text: W_i= tokenize(R_i'').
- Remove stopwords: $W_i' = W_i / Stopwords$

2. Embedding Generation

Generate dense vector embeddings $E = \{E_1, E_2, \dots, E_n\}$ using a pre-trained SBERT model:

$$E_i = SBERT(R_i')$$

Where $E_i \in \mathbb{R}^{384}$

3. Similarity Computation

Compute pairwise similarity between embeddings using the Euclidean distance.

$$d(E_i, E_j) = \sqrt{\sum_{k=1}^{384} (E_{i_k} - E_{j_k})^2}$$

where $E_{i_k} and\ E_{j_k}$ are the k^{th} dimensions of embeddings $E_i\ and\ E_j$.

4. Clustering

Perform hierarchical clustering using Agglomerative Clustering:

- Linkage method: Ward's linkage minimizes intra-cluster variance.
- Clustering criterion:

$$Merge\ clusters\ that\ minimize\ \Delta Variance$$

Result: Assign each requirement R_i to a cluster label C_i.

5. Evaluation

1. Cohesion: Measure intra-cluster similarity:

$$Cohesion = \frac{1}{|C_k|} \sum_{i,j \in C_k} Sim(E_i, E_j)$$

where $Sim(E_i, E_j) = 1 - d(E_i, E_j)$.

2. Separation: Measure inter-cluster dissimilarity:

$$Separation = \min_{i \in C_k, j \in C_l, k \neq l} d(E_i, E_j)$$

3. Silhouette Score: Combines cohesion and separation to measure the overall clustering quality:

$$Silhouette\ Score = \frac{b(i) - a(i)}{max(a(i), b(i))}$$

where $a(i)$ is the average intra-cluster distance and $b(i)$ is the average nearest-cluster distance.

4. Davies-Bouldin Index: Quantifies intra-cluster compactness and inter-cluster separation:

$$DB\ Index = \frac{1}{k} \sum_{i=1}^{k} \max_{j \neq i} \frac{\sigma_i + \sigma_j}{\mathrm{d}(C_i, C_j)}$$

Lower values indicate better clustering.

6. Complexity Analysis:

The complexity of different stages in the SQUIRE methodology is analyzed as follows:

1. Preprocessing Complexity (Time & Space):

Time Complexity: $O(n)$ where n is the number of requirements. Each requirement undergoes tokenization, stopword removal, and normalization, which operates linearly with respect to the number of requirements.

Space Complexity: $O(n)$, as each requirement is stored as a processed text sequence before embedding.

2. Embedding Generation Complexity (Time & Space):

Time Complexity: $O(n)$, since the Sentence-BERT (SBERT) model processes each requirement independently, leading to a linear complexity.

Space Complexity: $O(n \times d)$, where d is the embedding dimension (384 in the case of MiniLM-SBERT). The output matrix of embeddings requires storage proportional to the dataset size.

3. Clustering Complexity (Time & Space):

Time Complexity: $O(n^3)$ for Agglomerative Clustering, due to the hierarchical structure requiring pairwise distance computations and iterative merging.

4. Space Complexity: $O(n^2)$, as the clustering algorithm maintains a distance matrix for all requirement pairs.

This complexity analysis provided a clear distinction between time and space requirements at different stages of the methodology.

The SQUIRE algorithm efficiently organizes textual requirements into meaningful clusters by leveraging NLP embeddings and hierarchical clustering [28]. Its structured workflow ensures semantic precision and scalability while minimizing redundancy. With its foundation in advanced language models and practical clustering methods, SQUIRE lays the groundwork for a streamlined approach to handling complex requirements datasets, offering clarity and functionality to modern software engineering practices

IV. Results and Validation of Proposed Model

The validation of the proposed SQUIRE methodology was conducted using four real-world datasets sourced from a software company [32]. These datasets, representing diverse functional domains, included four different domains as shown in Table III. The primary objective of this validation was to assess the model's ability to accurately group semantically similar requirements into functional clusters.

TABLE III DATASETS OVERVIEW

Dataset	Domain	Number of Requirements
Attendance Management	Employee attendance tracking and reporting	26
E-commerce	Online shopping portal requirements	20
Lottery Management	Lottery system functionality	24
School Management	Educational institution operations	23

These datasets represent diverse functional requirements, providing a robust basis for testing the domain-agnostic capabilities of the SQUIRE methodology [32].

The evaluation focused on clustering outcomes, visualized through scatter plots and assessed quantitatively using Cohesion, Separation, Silhouette Score, and Davies-Bouldin Index. The results highlight the methodology's strengths and its performance across different datasets.

The clustering results for each dataset were visualized using Principal Component Analysis (PCA), reducing the 384-dimensional embeddings to two dimensions [23]. The scatter plots for the datasets Fig. 1 illustrate the semantic clusters, with each point representing a requirement and colors distinguishing the clusters.

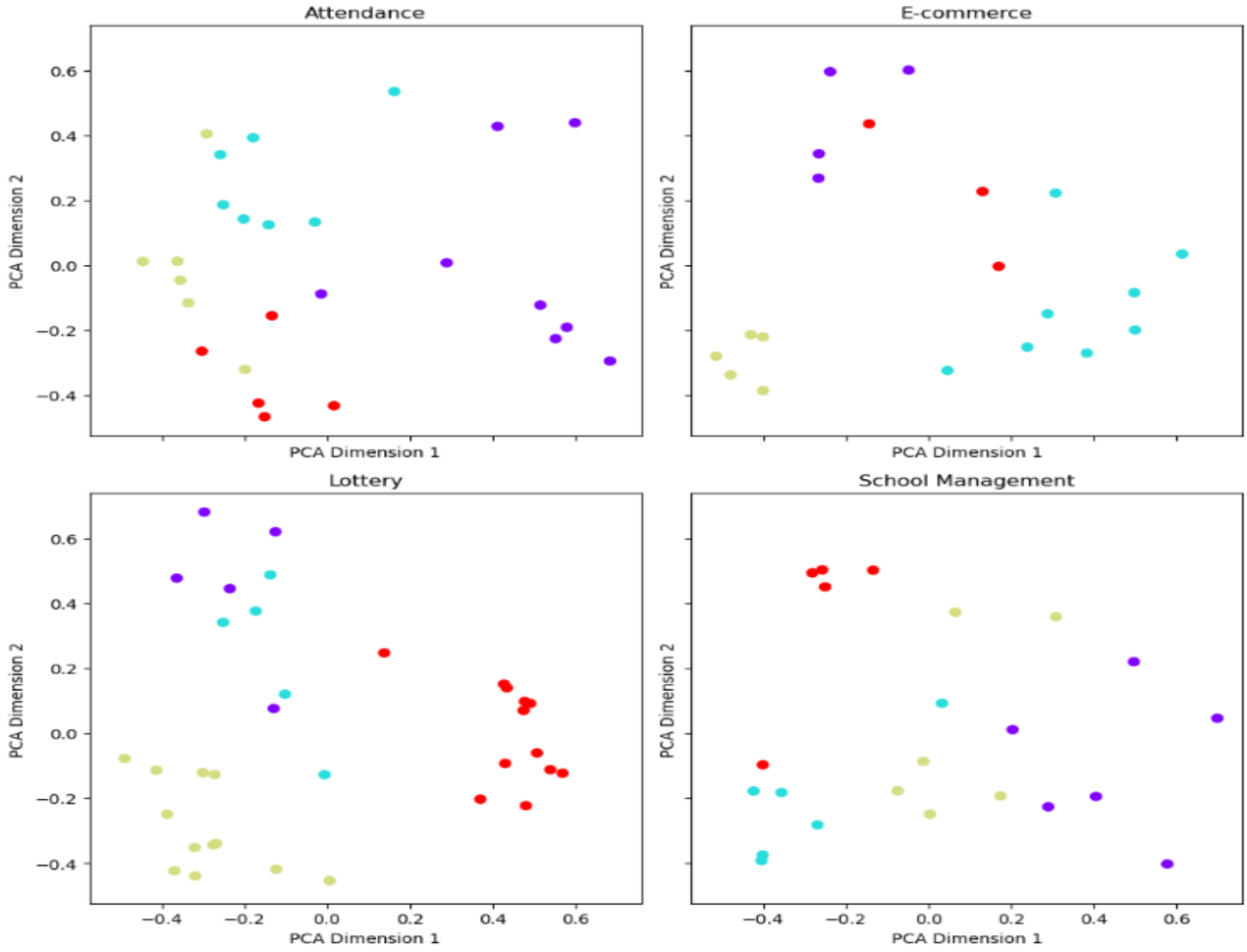

Fig. 1. Distribution of requirements clusters using PCA for each dataset.

The clustering performance is summarized in the Table IV below:

TABLE IV CLUSTERING METRICS FOR DIFFERENT DATASETS

Dataset	Cohesion	Separation	Silhouette Score	Davies-Bouldin Index
Attendance Management	0.2362	1.1306	0.1276	1.7027
E-commerce	0.1360	1.2911	0.1044	1.7195
Lottery Management	0.1096	1.2622	0.1587	1.8307
School Management	0.1940	1.1789	0.1174	1.7759

1) Cohesion: The Attendance Management dataset achieved the highest cohesion (0.2362), indicating well-grouped clusters. The Lottery Management dataset displayed slightly lower cohesion (0.1096), reflecting greater diversity within clusters.

2) Separation: The E-commerce dataset achieved the highest separation (1.2911), highlighting distinct clusters. The Attendance Management dataset had slightly lower separation (1.1306), possibly due to overlapping functional requirements.

3) Silhouette score: The Lottery Management dataset recorded the highest silhouette score (0.1587), indicating a good balance between cohesion and separation.

4) Davies-bouldin index: The Attendance Management dataset exhibited the lowest Davies-Bouldin Index (1.7027), reflecting compact and well-separated clusters. The Lottery Management dataset had the highest index (1.8307), suggesting room for improvement in cluster separation.

The validation results demonstrate the practical utility of the SQUIRE methodology in streamlining requirements engineering. By automating the clustering of semantically similar requirements, SQUIRE significantly reduces the manual effort required for organizing and analyzing requirements, allowing practitioners to focus on higher-value tasks such as decision-making and system design [7]. The methodology's ability to achieve high cohesion and separation across diverse datasets highlights its adaptability to different functional domains, making it a robust solution for handling large-scale and dynamic software projects. Furthermore, the integration of PCA-based visualizations enhances interpretability, providing clear insights into the relationships between requirements and supporting better traceability. Overall, SQUIRE offers a scalable and domain-agnostic approach that addresses critical challenges in requirements engineering, paving the way for more efficient and error-free software development processes.

V. DISCUSSION, LIMITATIONS AND FUTURE WORK

The validation results of the SQUIRE methodology demonstrate its effectiveness in clustering semantically similar requirements into distinct functional groups across diverse datasets. By leveraging Sentence-BERT embeddings and hierarchical clustering techniques, the methodology successfully addresses key challenges in requirements engineering, such as redundancy reduction, modularity, and enhanced traceability [2], [3], [13].

The methodology consistently produced meaningful clusters across four distinct datasets, sourced from a software company, representing domains Attendance Management, E-commerce, Lottery Management, and School Management [32]. These datasets varied in size and complexity, containing 20–26 requirements each, and provided a realistic foundation for testing the robustness and adaptability of the proposed approach. The clustering outputs revealed clear functional distinctions in well-structured domains like Attendance Management, while moderately overlapping clusters were observed in datasets like Lottery Management, which contained diverse and less structured requirements.

The evaluation metrics provided deeper insights into clustering performance:

Cohesion values indicated the strength of relationships within clusters, with the Attendance Management dataset achieving the highest cohesion, reflecting compact and meaningful clusters. Lower cohesion in the Lottery Management dataset suggests room for improvement in handling more heterogeneous requirements.

Separation metrics demonstrated the distinctiveness of clusters across datasets, with E-commerce showing the highest separation due to its well-defined functional boundaries.

Silhouette Score, a balance of cohesion and separation, highlighted the methodology's ability to achieve reasonable clustering quality across all datasets, with the highest score recorded for the Lottery Management dataset.

Davies-Bouldin Index values, indicative of clustering compactness and separation, were lowest for Attendance Management, reinforcing its strong cluster formations, while slightly higher values for Lottery Management reflected less compact clusters.

The visualizations further supported these findings, with distinct and well-separated clusters for structured datasets such as Attendance Management and School Management, while partially overlapping clusters were observed in Lottery Management due to functional overlaps in its requirements. The PCA-reduced scatter plots provide a clear representation of the semantic clustering process, aiding interpretability and further validating the methodology [14].

Despite the strong results, some limitations were observed. The clustering process relied on a fixed number of clusters, which may not always align with the inherent structure of the dataset. This could result in under- or over-clustering, especially in datasets with varied functional complexity. Additionally, the preprocessing pipeline, while robust, could be further enhanced with more domain-specific customizations, such as advanced synonym resolution or enhanced tokenization techniques, to address ambiguities in textual requirements. Finally, the methodology's reliance on static embeddings may limit its adaptability to rapidly evolving datasets, where requirements are frequently updated or redefined.

Future work will focus on enhancing the SQUIRE methodology by integrating advanced preprocessing techniques,

such as lemmatization and domain-specific synonym resolution, to improve the consistency and semantic accuracy of requirements. Adaptive clustering techniques, such as silhouette-based optimization, will be explored to dynamically determine the optimal number of clusters, ensuring better alignment with diverse datasets. Additionally, the use of more advanced models, such as GPT-based embeddings, will be investigated to capture deeper semantic relationships. A user-friendly tool incorporating real-time clustering, visualization, and traceability features will be developed to make the methodology more accessible to practitioners. Validation will be extended to real-world software engineering projects across various industries to evaluate practical applicability and scalability. Finally, interactive 3D visualizations and additional evaluation metrics will be introduced to improve interpretability and provide more comprehensive assessments of clustering quality.

VI. Conclusion

Requirements Engineering (RE) constitutes a pivotal phase in software development, focusing on the elicitation, definition, and management of stakeholder needs. Despite its criticality, traditional approaches frequently falter in managing the complexity, scale, and dynamism of contemporary software systems. Natural Language Processing (NLP) has emerged as a transformative enabler, offering automation in the analysis and organization of textual requirements. The SQUIRE framework, leveraging Sentence-BERT embeddings for semantic clustering, introduces a structured, scalable methodology for refining requirements management. By enhancing traceability, minimizing redundancy, and facilitating modular organization, SQUIRE addresses key inefficiencies in conventional RE practices. Its potential for broad applicability across diverse domains underscores its relevance to evolving software engineering demands. While SQUIRE has demonstrated efficacy, further refinements are necessary to optimize its alignment with stakeholder objectives and its adaptability to increasingly complex, dynamic requirements. Advancing the framework's flexibility and scalability will not only bridge theoretical innovations with practical application but also expand its impact across a wider spectrum of domains, establishing a robust foundation for next-generation RE methodologies.

References

[1] H. Villamizar, T. Escovedo and M. Kalinowski, "Requirements Engineering for Machine Learning: A Systematic Mapping Study," *2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, Palermo, Italy, 2021, pp. 29-36, doi: 10.1109/SEAA53835.2021.00013.

[2] L. Karlsson, Å. G. Dahlstedt, and A. Persson, “Requirements engineering challenges inmarket-driven software development An interview study withpractitioners”, Inf and Soft Techn, vol. 49, Dec. 2007, pp.588-604, doi: 10.1016/j.infsof.2007.02.008.

[3] R. Izhar, Kenneth Cosh, “Enhancing Agile Software Development: A Novel Approach to Automated Requirements Prioritization”, 2024 21st International Joint Conference on Computer Science and Software Engineering, 2024.

[4] Sonbol, R., Rebdawi, G., and Ghneim, N. (2022). The use of nlp-based text representation techniques to support requirement engineering tasks: A systematic mapping review. IEEE Access.

[5] Pei, Z., Liu, L., Wang, C., and Wang, J. (2022). Requirements engineering for machine learning: A review and reflection. In 2022 IEEE 30th International Requirements Engineering Conference Workshops (REW), pages 166–175. IEEE.

[6] Ahanger, M.M.; Wani, M.A.; Palade, V. sBERT: Parameter-Efficient Transformer-Based Deep Learning Model for Scientific Literature Classification. *Knowledge* **2024**, *4*, 397-421. https://doi.org/10.3390/knowledge4030022.

[7] Sehrish Alam, Shahid N. Bhatti,” Impact and challenges of requirement engineering in agile methodologies: A systematic review”, International Journal of Advanced Computer Science and Applications, 2017. http://dx.doi.org/10.14569/IJACSA.2017.080455.

[8] Radwan, Aya, et al. “An Approach for Requirements Engineering Analysis Using Conceptual Mapping in Healthcare Domain.” International Journal of Advanced Computer Science and Applications, vol. 12, no. 8, 2021, https://doi.org/10.14569/ijacsa.2021.0120822.

[9] Eyal Salman, H.; Hammad, M.; Seriai, A.-D.; Al-Sbou, A. Semantic Clustering of Functional Requirements Using Agglomerative Hierarchical Clustering. *Information* **2018**, *9*, 222. https://doi.org/10.3390/info9090222.

[10] del Sagrado, J., del Águila, I.M. Assisted requirements selection by clustering. *Requirements Eng* **26**, 167–184 (2021). https://doi.org/10.1007/s00766-020-00341-1.

[11] Bakar, N. H., et al. (2014). Tochs requirements reuse: Identifying similar requirements with latent semantic analysis and clustering algorithms. *International Journal of Software Engineering and Its Applications*.

[12] Das, S., Deb, N., Cortesi, A. et al. Sentence Embedding Models for Similarity Detection of Software Requirements. SN COMPUT. SCI. 2, 69 (2021). https://doi.org/10.1007/s42979-020-00427-1.

[13] Elhassan, H. et al. (2022) ‘Requirements Engineering: Conflict Detection Automation Using Machine Learning’, Intelligent Automation & Soft Computing, 33(1), pp. 259–273. Available at: https://doi.org/10.32604/iasc.2022.023750.

[14] R. Izhar, K. Cosh and S. N. Bhatti, "Enhancing Agile Software Development: A Novel Approach to Automated Requirements Prioritization," 2024 21st International Joint Conference on Computer Science and Software Engineering (JCSSE), Phuket, Thailand, 2024, pp. 286-293, doi: 10.1109/JCSSE61278.2024.10613648.

[15] M. P. Naik, H. B. Prajapati and V. K. Dabhi, "A survey on semantic document clustering,"2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), Coimbatore, India, 2015, pp. 1-10, doi: 10.1109/ICECCT.2015.7226036.

[16] Nagwani, N.K. Summarizing large text collection using topic modeling and clustering based on MapReduce framework. Journal of Big Data 2, 6 (2015). https://doi.org/10.1186/s40537-015-0020-5.

[17] Fougères, A.-J., & Ostrosi, E. (2020). Intelligent requirements engineering from natural language and their chaining toward CAD models. https://doi.org/10.48550/arXiv.2007.07825.

[18] Mehta, V., Agarwal, M. & Kaliyar, R.K. A comprehensive and analytical review of text clustering techniques. Int J Data Sci Anal 18, 239–258 (2024). https://doi.org/10.1007/s41060-024-00540-x.

[19] Haji, S.H., Jacksi, K., Salah, R.M. (2022). Systematic Review for Selecting Methods of Document Clustering on Semantic Similarity of Online Laboratories Repository. In: Daimi, K., Al Sadoon, A. (eds) Proceedings of the ICR’22 International Conference on Innovations in Computing Research. ICR 2022. Advances in Intelligent Systems and Computing, vol 1431. Springer, Cham. https://doi.org/10.1007/978-3-031-14054-9_23.

[20] Chen, D.; Wang, J. A Prompt Example Construction Method Based on Clustering and Semantic Similarity. Systems 2024, 12, 410. https://doi.org/10.3390/systems12100410.

[21] A. K. Singh, S. Mittal, P. Malhotra and Y. V. Srivastava, "Clustering Evaluation by Davies-Bouldin Index(DBI) in Cereal data using K-Means," *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)*, Erode, India, 2020, pp. 306-310, doi: 10.1109/ICCMC48092.2020.ICCMC-00057.

[22] Noor Hasrina Bakar, Zarinah M. Kasirun, Norsaremah Salleh, Feature extraction approaches from natural language requirements for reuse in software product lines: A systematic literature review, Journal of Systems

and Software, Volume 106, 2015, Pages 132-149, ISSN 0164-1212, https://doi.org/10.1016/j.jss.2015.05.006.

[23] R. Izhar, S. N. Bhatti, "Bridging Precision and Complexity: A Novel Machine Learning Approach for Ambiguity Detection in Software Requirements," in IEEE Access, vol. 13, pp. 12014-12031, 2025, doi: https://doi.org/10.1109/ACCESS.2025.3529943.

[24] A. Udomchaiporn, N. Prompoon and P. Kanongchaiyos, "Software Requirements Retrieval Using Use Case Terms and Structure Similarity Computation," 2006 13th Asia Pacific Software Engineering Conference (APSEC'06), Bangalore, India, 2006, pp. 113-120, doi: 10.1109/APSEC.2006.53.

[25] A. Radovanović, J. Li, J. V. Milanović, N. Milosavljević and R. Storchi, "Application of Agglomerative Hierarchical Clustering for Clustering of Time Series Data," 2020 IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe), The Hague, Netherlands, 2020, pp. 640-644, doi: 10.1109/ISGT-Europe47291.2020.9248759.

[26] Jie He, Wanqiu Long, and Deyi Xiong. 2022. Evaluating Discourse Cohesion in Pre-trained Language Models. In *Proceedings of the 3rd Workshop on Computational Approaches to Discourse*, pages 28–34, Gyeongju, Republic of Korea and Online. International Conference on Computational Linguistics.

[27] Hazem Abdelazim, Mohamed Tharwat and Ammar Mohamed, "Semantic Embeddings for Arabic Retrieval Augmented Generation (ARAG)" International Journal of Advanced Computer Science and Applications(IJACSA), 14(11), 2023. http://dx.doi.org/10.14569/IJACSA.2023.01411135.

[28] Murtagh, F., Legendre, P. Ward's Hierarchical Agglomerative Clustering Method: Which Algorithms Implement Ward's Criterion?. *J Classif* **31**, 274–295 (2014). https://doi.org/10.1007/s00357-014-9161-z.

[29] I. Dokmanic, R. Parhizkar, J. Ranieri and M. Vetterli, "Euclidean Distance Matrices: Essential theory, algorithms, and applications," in *IEEE Signal Processing Magazine*, vol. 32, no. 6, pp. 12-30, Nov. 2015, doi: 10.1109/MSP.2015.2398954.

[30] Amato, Alberto and Di Lecce, Vincenzo. "Data preprocessing impact on machine learning algorithm performance" *Open Computer Science*, vol. 13, no. 1, 2023, pp. 20220278. https://doi.org/10.1515/comp-2022-0278.

[31] Bin Wang and C.-C. Jay Kuo. 2020. SBERT-WK: A Sentence Embedding Method by Dissecting BERT-Based Word Models. IEEE/ACM Trans. Audio, Speech and Lang. Proc. 28 (2020), 2146–2157. https://doi.org/10.1109/TASLP.2020.3008390.

[32] rahat-23. "GitHub - Rahat-23/Datasets-For-Requirements." *GitHub*, 2024, github.com/rahat-23/Datasets-for-Requirements.