Extraction of Class Candidates from Scenario in Software Requirements Specifications

Rasi Aziizah Andrahsmara[1], Daniel Oranova Siahaan[2]
[1,2] Departement of Informatics Engineering, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia
[1] rasiaziza@gmail.com
[2] daniel@if.its.ac.id

Accepted on July 08th, 2025
Approved on July 24th, 2025

***Abstract*— The development of a software application involves a comprehensive process of defining and documenting software requirements. Among various modeling activities, sequence diagrams serve a vital role in illustrating dynamic interactions among system components. However, manually constructing these diagrams from natural language Software Requirements Specifications (SRS) is often labor-intensive, inconsistent, and prone to human error, especially when the text is complex and unstructured. This study focuses on automating the extraction of candidate elements specifically classes, subclasses, and attributes from the scenario sections of SRS documents. These sections are typically written in narrative form and are rich in behavioral information. The proposed method integrates Natural Language Processing (NLP) using Bidirectional Encoder Representations from Transformers (BERT) for contextual embeddings and a Support Vector Machine (SVM) classifier to categorize each noun phrase accordingly. Two datasets, SIData and SILo, with distinct domain characteristics and writing styles, were used to evaluate the system's performance. While the system demonstrates the feasibility of the approach in identifying relevant elements, limitations such as low precision and false-positive rates highlight the need for further refinement in classification accuracy, generalizability, and semantic understanding of entity relationships. These challenges present opportunities for future work, including improvements in preprocessing strategies, data augmentation, and the use of ontologies for domain-specific consistency.**

Index Terms—*Software Requirement Specifications; Class Diagram; Sequence Diagram; Class Extraction; UML; Natural Language Processing; SVM; Scenario; Scenario in SRS;*

I. INTRODUCTION

The Unified Modeling Language (UML) has long been used as a standard way to describe how software systems are structured and how they behave. As noted by the Object Management Group (OMG), UML acts as a general-purpose visual language that helps various stakeholders understand and communicate the system's design elements. Among the many diagrams in UML, class diagrams are typically used to show the system's static structure, while sequence diagrams focus more on how components interact over time. These diagrams are valuable not only for communication but also for documenting and maintaining complex systems throughout the development lifecycle.

In real-world software engineering, the Software Requirements Specification (SRS) is one of the most crucial documents [1]. It outlines what the system should be able to do covering both functional and non-functional aspects. Often, the SRS contains narrative scenarios to help convey system behavior in a more intuitive way. Converting these narratives into structured diagrams such as class or sequence diagrams is essential, but doing it manually can be time-consuming, prone to errors, and difficult to maintain especially when the system grows more complex [2]. method for classifying sentences in Software Requirements Specifications (SRS) using Natural Language Processing (NLP) techniques and BERT embeddings. Their work highlights the effectiveness of deep contextual representation in improving the understanding of requirement-related sentences in software documentation [3].

Several researchers have tried to address this problem. For example, Yang and Sahraoui [4] highlighted how tricky it is to convert natural language into UML because of the ambiguity and inconsistency of human language. Others, like Shweta et al [5], began experimenting with transformer models to get better at identifying diagram components in textual requirements. Malik et al. [6] used BERT to pick out specific entities from SRS documents, and Ferrari et al. [7] looked into how large language models (LLMs) could help generate sequence diagrams automatically. These efforts show that there's growing interest in using Natural Language Processing (NLP) to simplify software modeling. [8] Even so, a number of challenges remain. One of the biggest gaps is in extracting fine-grained elements like deciding whether a noun phrase refers to a class, a subclass, or just an attribute. This becomes even more difficult when the system relies on scenario sections, which are usually written in free-form narratives. These parts can be rich in context but hard for machines to understand without deeper language processing.

That's where this study comes in. We propose a method to automatically extract useful UML entities specifically candidate classes, subclasses, and attributes from the scenario sections in SRS documents. To do this, we combine the power of BERT embeddings with a Support Vector Machine (SVM) classifier that assigns each noun phrase to the right category based on its context [9]. The idea is to create a rough structure that can later support the construction of sequence diagrams.

To see how well this method works, we tested it on two different datasets: SIData and SILo. The SIData and SILo datasets used in this study were specifically selected due to their completeness and availability of both Software Requirements Specifications (SRS) and corresponding class diagrams, which are essential for evaluation. The SIData dataset originates from an internal project related to environmental assessment systems, while SILo is derived from a logistics information system used in academic environments. Both datasets were obtained from the Department of Informatics, Institut Teknologi Sepuluh Nopember (ITS), Indonesia. The SIData dataset has more technical and domain-specific language, while the SILo dataset uses a more narrative tone, similar to how people describe scenarios in real-life situations. This contrast helps us see how flexible and reliable our method is across different writing styles. In our evaluation, we used common metrics like precision, recall, and F1-score to measure how accurately the system could identify and classify the elements. The results gave us useful insights into where the system performs well and where it still needs improvement.

II. Methodology

This section outlines the methodological framework employed to automatically extract class-related elements from scenario-based Software Requirements Specifications (SRS). The process consists of several sequential steps, beginning with the extraction of noun phrases from the scenario text using BERT embeddings, followed by classification of these phrases into class candidates. The extracted elements are then compiled and compared with ground truth data derived from corresponding class diagrams to evaluate the system's accuracy. Each step is designed to ensure a systematic and replicable approach for validating the effectiveness of the proposed method as can be seen in Fig 1.

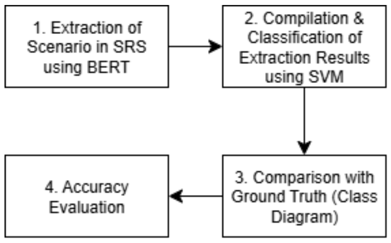
1. Extraction of Scenario in SRS using BERT
2. Compilation & Classification of Extraction Results using SVM
3. Comparison with Ground Truth (Class Diagram)
4. Accuracy Evaluation

Fig. 1. Methodology

A. *Extraction of Scenario in SRS Scenario using BERT*

This stage focuses on capturing user interaction flows with the system as described in the functional requirements. Specifically, the research concentrates on processing only the scenario sections of SRS documents, as these are considered the most informative for identifying critical entities namely, classes, subclasses, and attributes which will later form part of the class diagram model [10].

The rationale behind focusing on scenarios stems from prior studies, which have shown that scenario narratives often encapsulate rich behavioral information essential for generating models such as sequence or class diagrams. In this study, the dataset used is stored in plain text (.txt) format. One example is a scenario excerpt from the SIData dataset, as shown in Fig. 2.

To extract relevant information, the text undergoes several Natural Language Processing (NLP) steps. These include noun phrase identification using the spaCy library [11], followed by embedding representations using BERT. These embeddings are then classified into predefined categories (class, subclass, attribute) using a Support Vector Machine (SVM) classifier. This hybrid technique aligns with recent advancements in contextual entity recognition, which have demonstrated high performance in extracting domain-relevant entities from unstructured documents. Implemented BERT models to classify of contextual embeddings and sequential modeling can yield strong classification performance on sentiment-based datasets [12]. In a comparative analysis, examined the influence of different pre-trained models on the accuracy of BERT for text classification, emphasizing the importance of selecting suitable base models to optimize performance in domain-specific tasks [13].

The extracted entities are then evaluated by comparing them with the reference class diagrams, which have been manually constructed and serve as the ground truth for this experiment.

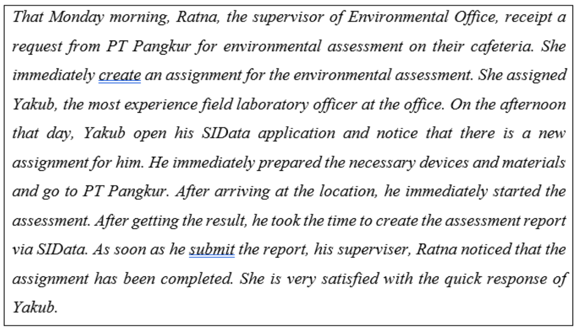
That Monday morning, Ratna, the supervisor of Environmental Office, receipt a request from PT Pangkur for environmental assessment on their cafeteria. She immediately create an assignment for the environmental assessment. She assigned Yakub, the most experience field laboratory officer at the office. On the afternoon that day, Yakub open his SIData application and notice that there is a new assignment for him. He immediately prepared the necessary devices and materials and go to PT Pangkur. After arriving at the location, he immediately started the assessment. After getting the result, he took the time to create the assessment report via SIData. As soon as he submit the report, his superviser, Ratna noticed that the assignment has been completed. She is very satisfied with the quick response of Yakub.

Fig. 2. Scenario of SIData

B. *Compilation and Classification of Extraction Results using SVM*

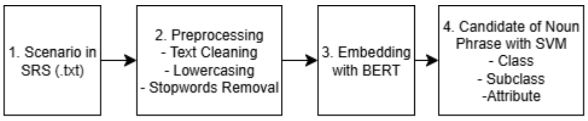
1. Scenario in SRS (.txt)
2. Preprocessing - Text Cleaning - Lowercasing - Stopwords Removal
3. Embedding with BERT
4. Candidate of Noun Phrase with SVM - Class - Subclass -Attribute

Fig. 3. Extraction Process

The preprocessing phase starts by cleaning the raw text to remove non-alphabetic characters, converting all words to lowercase, and eliminating stopwords. These are common techniques in Natural Language Processing (NLP) workflows, aimed at minimizing noise and improving the performance of later processing steps . After the text is cleaned, noun phrases are identified using spaCy's built-in linguistic features, which rely on part-of-speech (POS) tagging and syntactic dependency parsing to detect meaningful noun chunks.

Next, each identified noun phrase is transformed into a contextual embedding using a BERT model that has been fine-tuned specifically for the task of entity classification in software requirement texts. This fine-tuning process involves training BERT on manually annotated scenario data from SRS documents and appending a classification layer to label each noun phrase as a class, subclass, or attribute [14].

To improve categorization accuracy, these BERT-generated embeddings are then passed to a Support Vector Machine (SVM) classifier. The process consists of several sequential steps, beginning with the extraction of noun phrases from the scenario text using BERT embeddings. These embeddings are then fed into a Support Vector Machine (SVM) classifier to distinguish between valid and non-valid class candidates, as illustrated in Fig. 3. The extracted elements are then compiled and compared with ground truth data derived from corresponding class diagrams to evaluate the system's accuracy. Each step is designed to ensure a systematic and replicable approach for validating the effectiveness of the proposed method. This hybrid method leveraging deep contextual understanding from BERT alongside the robustness of SVM decision boundaries has been shown in previous studies to yield reliable results in entity classification tasks [15]. Support Vector Machine (SVM) approach to classif. The study confirmed the robustness of SVM in handling tasks with limited training data [16].

C. *Comparison with Ground Truth*

To assess the effectiveness of the proposed extraction method, an evaluation was conducted by comparing the extracted entities with a manually constructed class diagram, which serves as the ground truth. This comparison focuses on identifying matches between the predicted and actual elements, including class names, attributes, and subclass hierarchies.

To facilitate automated comparison, the reference class diagram was converted into a structured JSON format, enabling consistent parsing and element-wise alignment . The evaluation process then involves checking for the presence or absence of each predicted entity within the reference diagram.

Through this approach, the system's performance is quantitatively assessed, providing insight into its accuracy and relevance in identifying meaningful entities. Metrics such as precision, recall, and F1-score are used to measure how well the extracted results align with the expected outputs, thereby reflecting the practical applicability of the method in real-world software modeling tasks.

D. *Accuracy Evaluation*

The performance of the proposed extraction method is evaluated using a set of well-established metrics: accuracy**,** precision**,** recall**,** and F1-score. quantifies how many of the elements identified by the system are actually correct, while recall measures how many of the relevant elements present in the ground truth were successfully detected. The F1-score, serving as the harmonic mean of precision and recall, provides a balanced indicator of the system's ability to minimize both false positives and false negatives.

Beyond these core metrics, a descriptive statistical analysis is also performed to gain deeper insight into the model's performance. This includes calculating the minimum**,** maximum**,** mean**,** and standard deviation for each entity category namely class, subclass, and attribute [17]. The minimum and maximum values indicate the range of model performance, highlighting its best and worst outcomes across the evaluation. The mean reflects the central tendency, offering a general impression of accuracy across multiple test cases [18]. Meanwhile, the standard deviation captures the degree of performance variability, providing an indirect measure of the model's consistency and reliability when applied to datasets with differing linguistic characteristics.

Table 1. Confusion Matrix for Entity Extraction

Dataset	True Positive	False Positive	False Negative
SIData	2	27	18
SILo	3	24	9

The information extraction process using the BERT and SVM approach revealed that the overall performance of the system could not yet be classified as satisfactory. This is reflected in the evaluation metrics precision, recall, and F1-score which tend to be relatively low. Based on the evaluation results, the obtained performance metrics indicate the extent to which the system succeeded in accurately extracting information from the SRS documents, as presented in Table 2.

Table 2. Performance Evaluation

Dataset	Accuracy	Precision	Recall	F1-Score
SIData	4.2	0.069	0.10	0.250
SILo	8.3	0.111	0.25	0,157

To gain deeper insights into the classification performance of the BERT and SVM model across two datasets SIData and SILo a visual evaluation was conducted using a whisker plot. This diagram provides an intuitive overview of the distribution of three key metrics: precision, recall, and F1-score. Powers argues that common metrics (precision, recall, F1-score) can be misleading without understanding underlying biases. It explores alternative metrics like informedness and Markedness for a more principled evaluation [19]. This consolidates common performance metrics for classification and provides guidance on when to apply statistical significance testing [20].

For the SIData dataset, the plot reveals a tightly clustered boxplot within the lower range (below 0.10), highlighting the model's overall poor performance in identifying and categorizing entities accurately. Specifically, the model achieved a precision of 0.069, recall of 0.100, and F1-score of 0.082. The short whiskers in Figure 5 further reinforce this outcome, indicating low variance and minimal dispersion among prediction results suggesting that most classification attempts consistently performed poorly.

In contrast, the SILo dataset exhibits a broader distribution and noticeably improved metric values. The model attained a precision of 0.111, recall of 0.250, and F1-score of 0.157, with the whisker plot showing longer whiskers and a wider interquartile range. This implies greater variability in the model's performance and a modest improvement in its ability to generalize across different types of textual structures. Although the overall precision remains relatively low, the wider distribution reflects the model's potential to more effectively capture relevant phrases in less rigid, narrative-style documents such as those in the SILo dataset.

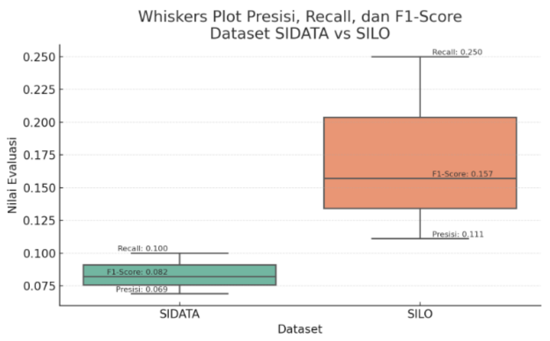

Fig. 5. Whisker Plot of Dataset

The observed differences in performance distribution between the two datasets can be attributed to their inherent linguistic characteristics. The SIData dataset generally features more informal and context-dependent phrasing, which introduces ambiguity and challenges in accurately identifying and classifying entities. In contrast, the SILo dataset is composed of more structured and repetitive terminology, often found in formal technical documentation, which aids in improving classification consistency. Furthermore, disparities in the total number of noun phrases, as well as the syntactic clarity of those phrases, play a significant role in influencing the classification performance across both datasets.

III. Result and Discussion

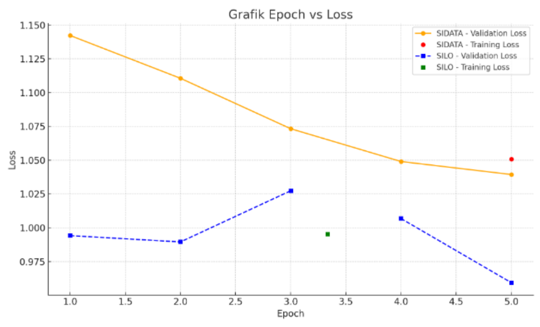

Fig. 6. Training loss of Dataset

The figure above illustrates a comparative analysis of training and validation loss curves observed during the BERT fine-tuning process across several epochs, using the SIData and SILo datasets. The orange line depicts the validation loss for the SIData dataset, which demonstrates a steady decline from epoch 1 through epoch 5. This downward trend suggests a stable and consistent learning process. The red dot, positioned at epoch 5, represents the training loss at that stage, providing a benchmark for evaluating how well the model has learned.

On the other hand, the blue line, which corresponds to the SILo dataset, shows a more erratic validation loss trajectory. Notably, there is a temporary increase in validation loss at epoch 3 before it eventually decreases to its lowest point at epoch 5. The green dots indicate training loss values captured at various epochs, offering further insight into the model’s convergence behavior. This fluctuation suggests that the SILo dataset presents greater variability or complexity, which may challenge the model's ability to consistently internalize training patterns.

Table 3. Training Log of SIData

Epoch	Iteration	Training Loss	Validation Loss
1.0	2	-	1.1422637701034546
2.0	4	-	1.1105989217758179
3.0	6	-	1.0732616186141968
4.0	8	-	1.048999309539795
5.0	10	1.0507	-
5.0	10	-	1.0393873453140259

Table 4. Training Log of SILo

Epoch	Iteration	Training Loss	Validation Loss
1.0	3	-	0.9941515326499939
2.0	6	-	0.9894852042198181
3.0	9	-	1.0273746252059937
3.3335	10	0.9951	-
4.0	12	-	1.0067888498306274
5.0	15	-	0.9593325257301331

Tables 3 and 4 display the detailed training logs of the BERT fine-tuning process for both SIData and SILo datasets, including the training loss and validation loss values across various epochs and iterations. For SIData (Table 3), the validation loss shows a consistent decline from epoch 1.0 to epoch 5.0, indicating a steady learning curve. The lowest validation loss, 1.039837435140259, was achieved at epoch 5.0 with 10 iterations, while the only recorded training loss value was 1.0507 at epoch 5.0 with 1 iteration.

In contrast, Table 4 illustrates the SILo dataset’s training progression, where validation loss exhibits more fluctuation. Despite a slight increase at epoch 3.0, the validation loss eventually decreased to its lowest value of 0.9593325257301331 at epoch 5.0 with 15 iterations. The training loss was recorded at epoch 3.3335, reaching 0.9951. These variations reinforce the earlier observation that SILo’s data complexity affects the stability and convergence of the model during training.

In this evaluation, the program demonstrated a significant dependence on the characteristics of the dataset. For the SILo dataset, which contains more natural and narrative-style text, the system exhibited better performance in information extraction. Conversely, for the SIData dataset, which is more formal and includes many technical terminologies, the system’s performance declined considerably. This indicates that the system’s generalization capability across various document styles is still limited and requires further improvement through increased training data variation and more refined classification methods. These results suggest that the model is better at identifying entities in datasets with certain structural and editorial patterns, though enhancements are needed both in the labeling process and in the classification architecture to ensure more stable and accurate performance across diverse data types.

Based on two scenario-based tests using different datasets, there is a clear tendency for the system to perform better when the data exhibits a more natural and explicit structure. This is evident from the stronger results observed on the SILo dataset, which utilizes operational descriptive narratives, as opposed to the SIData dataset that employs a more rigid and technically formatted language. The model tends to extract entities more effectively from sentences resembling everyday human communication. For instance, in the SILo dataset example: “After completing the payment, Ratna printed a receipt and handed it over to Yakub. ... Ratna sent Yakub to the warehouse. Yakub immediately went to the warehouse,” the system could accurately extract entities such as *Yakub*, *receipt*, and *warehouse* due to the narrative structure. In contrast, a sentence like “AssignmentForm consists of various attributes such as SamplingID, TestingDevice, and SampleType which are stored in MasterData” from the SIData dataset proved more difficult to process due to its dense, technical form and lack of explicit actor interactions.

Such discrepancies contribute to unstable system performance, as reflected in the fluctuating precision, recall, and F1-Score values across the two datasets. It is crucial to recognize that the alignment between extracted results and ground truth data does not rely solely on the presence of entity names (e.g., attributes), but also on the contextual accuracy specifically, the correct class hierarchy to which those elements belong. For example, while the attribute *invoiceNumber* may appear in the extraction output, if it is assigned to the class *Payment* or *Receipt* instead of *Invoice* (as defined in the reference data), the system fails to classify it correctly. Therefore, despite being lexically correct, such attributes are considered contextually invalid and are treated as False Negatives (FN) in the evaluation.

Similar misclassifications were observed for attributes such as *status* and *supplierName*, where the system extracted the correct term but linked it to the wrong class. Even when such attributes are extracted, if they do not match their intended contextual placement in the reference class structure, they are still categorized as classification errors contributing to FN rates.

During evaluation, anomalies were identified where entities matching the ground truth terminology were extracted but placed under incorrect class contexts. Examples include entities like *invoiceNumber* and *status*, which were correctly recognized but misclassified into inappropriate categories. These misalignments resulted in the entities being counted as False Negatives, indicating a deeper issue in the system’s understanding of semantic relationships and hierarchical structures among components in the document. This highlights the ongoing challenge of enabling the system to comprehend and preserve the semantic and structural fidelity expected in UML-based document analysis.

IV. Conslusion

This study proposed a method to automatically extract conceptual elements namely classes (object), subclasses, and attributes from scenario-based Software Requirements Specification (SRS) documents. The process involved several stages, including text preprocessing, noun phrase extraction using spaCy, contextual embedding using BERT, and classification with an SVM classifier. The extracted candidates were then compared with the ground truth in class diagrams to evaluate structural alignment and accuracy. Overall, the method demonstrates the feasibility of integrating NLP and machine learning techniques for supporting early-stage software design automation.

The result of the extraction process consists of conceptual elements such as objects, subclasses, and attributes, which are then compared with the reference structure in the class diagram to measure their alignment and accuracy. This research process began with the collection and conversion of datasets in the form of Software Requirements Specification (SRS) documents into a compatible format, followed by text pre-processing stages such as the removal of non-alphabetic characters, lowercasing, and stopword elimination. After the text cleaning phase, noun phrase extraction was carried out using the spaCy NLP model, which was subsequently processed using BERT vector representations and classified using an SVM classifier to map entities into classes, subclasses, and attributes. Qualitatively, the primary objective of this study was achieved, which is the development of a framework capable of processing SRS documents and producing outputs that can be directly compared. The developed program has successfully identified several entities from the text and classified them into categories of class, subclass, or attribute, albeit with limited precision. Quantitatively, the system performance was evaluated using precision, recall, and F1-Score metrics on two different datasets: SIData and SILo. The results show that the precision values ranged from 8.3% to 16.1%, with an average of 12.2%. Recall values ranged from 9.5% to 38.5%, with an average of 24%. The F1-Score, which reflects the balance between precision and recall, ranged from 8.9% to 22.9%, with an average value of 15.9%. These values indicate that, although the system has functioned according to its intended purpose, the accuracy and relevance of the extraction results still require significant improvement for practical application in software development.

Overall, the performance of the extraction system remains far from optimal, with low accuracy rates across both datasets. The model tends to produce high false positive (FP) rates and still fails to recognize a significant portion of entities present in the reference data.

For further study, a few specific improvements may help strengthen the method. One is refining the preprocessing step to handle camelcase terms, since these often include multiple meaningful parts. It may also help to include linguistic cues, such as Part-of-Speech (POS) tags or the position of a noun phrase in a sentence, which can guide more accurate classification. Adding training data through paraphrased or reworded sentences could improve generalization. Using domain specific glossaries or ontologies might also support better consistency when dealing with specialized terms. Finally, future evaluations should consider reporting performance separately for classes, subclasses, and attributes to better understand where the model performs well or struggles.

References

[1] A. Ferrari, “From Natural Language Requirements to Sequence Diagrams via Large Language Models,” *Proc. 31st IEEE Int. Requirements Engineering Conf. (RE’23)*, IEEE, pp. 220–231, 2023. doi: 10.1109/RE57524.2023.00032.

[2] A. Ferrari, "Model Generation with LLMs: From Requirements to UML Sequence Diagrams," in *IEEE*, DOI: 10.1109/REW61692.2024.00044, 2024.

[3] H. Casanova, "BERT_SE: A Pre-trained Language Representation Model for Software Engineering," *ULTIMA InfoSys: Jurnal Ilmu Sistem Informasi*, vol. 14, no. 2, pp. 139–148, 2021. doi: 10.31937/si.v14i2.1750.

[4] H. Yang, "Towards Automatically Extracting UML Class Diagrams from Natural Language Specifications," in *Proceedings of the 27th IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, London, ON, Canada, pp. 548–552, 2020. doi: 10.1109/SANER48235.2020.9055078

[5] S. Shweta, "Advancing Class Diagram Extraction from Requirement Text: A Transformer Based Approach," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, New Delhi, India, 2023. doi: 10.18653/v1/2023.acl-long.872

[6] M. Malik, "BERT Based Domain Entity Recognition in Software Requirements," *IEEE Access*, vol. 9, pp. 73096–73105, 2021. doi: 10.1109/ACCESS.2021.3081234

[7] G. Kmetty, R. Gulyás, and A. Nyisztor, "Boosting classification reliability of NLP transformer models," *Information Processing & Management*, vol. 60, no. 2, p. 103254, 2023. doi: 10.1016/j.ipm.2022.103254

[8] P. K. Mahajan and R. Mahajan, "Information extraction using NLP techniques," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 5, pp. 1949–1957, 2022. doi: 10.1016/j.jksuci.2020.05.004

[9] S. Yang and H. Sahraoui, "Towards Automatically Extracting UML Class Diagrams from Natural Language Specifications," in *Proceedings of the 2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, London, ON, Canada, 2022, pp. 573–577. doi: 10.1109/SANER48235.2020.9055078

[10] R. M. Putra and N. Yudistira, “Extractive Text Summarization Using BERT-Based Model on Bahasa

Indonesia Scientific Articles," *ULTIMA InfoSys: Jurnal Ilmu Sistem Informasi*, vol. 14, no. 2, pp. 128–138, 2023. doi: 10.31937/si.v14i2.3021

[11] N. Yudistira, "Analisis Pengaruh Pre-Trained Model Terhadap Akurasi Model BERT Untuk Klasifikasi Teks," *ULTIMA InfoSys: Jurnal Ilmu Sistem Informasi*, vol. 13, no. 2, p. 98–106, 2022.

[12] H. Zhaou, "A Survey of Deep Learning Approaches for NLP," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 11, no. 5, Article 49, pp. 1–41, 2020. doi: 10.1145/3383316

[13] N. Yudistira, "Penerapan Support Vector Machine dalam Klasifikasi Emosi pada Ulasan Produk," *ULTIMA InfoSys: Jurnal Ilmu Sistem Informasi*, vol. 112–120, no. 2, p. 12, 2021.

[14] Y. Li, "Automatic UML Class Diagram Generation from Natural Language Requirements Using BERT and Graph Convolutional Networks," *IEEE Access*, vol. 10, pp. 31207–31219, 2022. doi: 10.1109/ACCESS.2022.3156205

[15] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. of the 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, vol. 1, pp. 4171–4186, 2019. doi: 10.18653/v1/N19-1423

[16] T. Wolf *et al.*, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP)*, pp. 38–45, 2020. doi: 10.18653/v1/2020.emnlp-demos.6

[17] Yildirim, "Adaptive Fine-tuning for Multiclass Classification over Software Requirement Data," 2023.

[18] T. Okuda, A. Okada, and Y. Morikawa, *"Transformation Method from Scenario to Sequence Diagram,"* in *Proceedings of the 10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K)*, vol. 1: KDIR, pp. 136–143, 2018.

[19] D. M. W. Powers, "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation," *arXiv preprint*, Oct. 2020. doi:10.48550/arXiv.2010.16061.

[20] A. Author *et al.*, "Evaluation metrics and statistical tests for machine learning," *Scientific Reports*, vol. 13, article 9821, May 2023. doi:10.1038/s41598-024-56706-x.