LLM Token Consumption Optimization Using BERT Pre-filtering and Retrieval-Augmented Generation (RAG)

Nanang Kurnia Alfian[1], Robert Marco[2]
Master of Informatics Engineering, Universitas Amikom Yogyakarta

The adoption of Large Language Models (LLM) for software issue resolution causes token consumption inefficiency due to noisy and redundant context. This research proposes a hybrid system placing a BERT classification model as a context relevance pre-filter upstream of the Retrieval-Augmented Generation (RAG) pipeline, with hyperparameters optimized using Bayesian Optimization. A Design Science Research approach is applied through an ablation study of five scenarios (S0-S4) on GitHub Issues and SWE-bench datasets. The classification model achieves an F1-macro of 0.887, and BERT-based filtering (S2) reduces token consumption by 8.35% versus baseline, statistically significant (Wilcoxon, $p<0.05$). Bayesian Optimization yields a validation F1-macro of 0.889, higher than default hyperparameters. The main finding: token efficiency comes from the BERT pre-filter component, not from RAG which adds context.

Keywords: Token Optimization, BERT, RAG, Bayesian Optimization, Text Classification.

This is an open access article under theCC BY-NClicense

Corresponding Author:
Nanang Kurnia Alfian
Master of Informatics Engineering, Universitas Amikom Yogyakarta
n.kurnia@students.amikom.ac.id

1. Introduction

The rapid development of Large Language Models (LLMs) has transformed the landscape of software development. The Stack Overflow Developer Survey reported that 84% of developers have adopted or plan to adopt artificial intelligence-based tools. One of their applications is assisting developers in understanding and resolving issues within software projects. However, commercial LLM services are typically priced based on the number of processed tokens, making token consumption efficiency a direct determinant of operational costs.

Several approaches have been proposed to reduce token consumption. Retrieval-Augmented Generation (RAG) retrieves relevant documents from a knowledge base [1], [14]; BERT-based text classification automatically evaluates the relevance of text segments [2]; and Bayesian Optimization efficiently tunes model hyperparameters [3]. Nevertheless, each method has limitations. Conventional RAG lacks a pre-retrieval filtering mechanism, BERT is highly sensitive to hyperparameter settings, and manual tuning or Grid Search requires substantial computational resources.

These limitations become particularly evident in software issue resolution tasks. GitHub Issues often contain considerable contextual noise [4], while the lost-in-the-middle phenomenon [5] reduces an LLM's ability to effectively utilize the most important information. This problem is further exacerbated by conventional RAG systems, which, without a pre-retrieval filtering mechanism, may actually increase token consumption by retrieving and incorporating irrelevant documents.

Based on the literature review presented in Section 2, BERT, RAG, and Bayesian Optimization have each been investigated independently. However, no previous study has positioned BERT as a relevance pre-filter at the upstream stage of the RAG pipeline while simultaneously optimizing the classifier's hyperparameters using Bayesian Optimization for software issue resolution tasks. This research gap forms the basis of the

present study's contributions: (1) integrating BERT as a relevance pre-filter within a RAG pipeline for the software engineering domain, an approach that has not previously been explored; (2) extending the application of Bayesian Optimization from prompt optimization [6] to classification model hyperparameter optimization; and (3) providing quantitative empirical evidence of the individual contribution of each component to token efficiency through an ablation study.

2. Literature Review

This section reviews previous studies related to the three main components of this research—BERT, Retrieval-Augmented Generation (RAG), and Bayesian Optimization—and positions the contribution of the proposed approach in addressing the limitations of each method, as summarized in Table 1.

Table 1. Comparison of Previous Studies

Study	Method	Research Gap Addressed
Sabbatella [6]	Bayesian Optimization (BO) for LLM Prompt Optimization	Limited to prompt optimization rather than classification models
Lewis [1]	Retrieval-Augmented Generation (RAG) (DPR + BART)	No pre-retrieval filtering mechanism
Liu [5]	*Lost-in-the-Middle* Study	Analytical study without an automated filtering solution
Devlin [2]	Bidirectional BERT Encoder	Not yet applied as a pre-filtering mechanism for RAG context
Akiba [3]	Optuna Bayesian Optimization (TPE)	Has not been investigated in the software engineering domain
Jimenez [4]	SWE-bench Benchmark	Focuses on code-solving performance rather than token efficiency

Based on Table 1, this research positions itself as an integration of three methods—BERT as a pre-filter, RAG as a retrieval component, and Bayesian Optimization as a hyperparameter tuner—which have never been combined in the domain of solving software problems with a focus on token efficiency.

The theoretical foundation includes the Transformer architecture with a self-attention mechanism [7] as the basis for BERT, a pre-trained language model optimized using parameter-efficient techniques [8] and low-cost adaptation [9], and reasoning techniques such as chain-of-thought [15]. Bayesian Optimization as a sample-efficient black-box optimization method is comprehensively discussed by Archetti and Candelieri [10], with modern implementations such as BoTorch [11]. The sentence embedding representation for retrieval refers to Sentence-BERT [12].

3. Methods

This research is an experimental quantitative study using a Design Science Research approach [13]. Two public datasets were used: GitHub Issues (5,257 issues from Python repositories) and SWE-bench [4] (2,294 instances). A total of 24,185 context segments were generated from the GitHub Issues and SWE-bench datasets. Relevance labeling was performed using a combination of rule-based methods, TF-IDF, and manual validation. A context segment was categorized as relevant when the information contained in the segment was related to the corresponding software issue and provided contextual information that could support the understanding or resolution of the issue. Conversely, a segment was categorized as not relevant when its content was unrelated to the corresponding issue or did not provide useful contextual information for addressing the issue. To verify the generated labels, manual validation was conducted on a

15% sample of the context segments. The manually validated samples were used to check the consistency of the automatically generated relevance labels before the dataset was used for training the BERT-based context classification model. The final dataset consisted of 55.5% relevant and 44.5% not-relevant segments and was divided into training, validation, and test sets using an 80:10:10 ratio with a fixed random seed of 42. A limitation of the dataset construction is that only 15% of the automatically generated labels were manually validated, and the validation did not involve multiple independent annotators; consequently, inter-annotator agreement could not be assessed.

The bert-base-uncased BertContextClassifier classification model received input in the format [CLS] segment [SEP] query [SEP]; the token representation [CLS] was passed to a dropout layer and a linear classifier (768 to 2). Weighted cross-entropy was used to handle class imbalance. Hyperparameters were optimized using Bayesian Optimization (Tree-structured Parzen Estimator algorithm) to maximize the F1-macro validation. The RAG pipeline is built with a text splitter (chunk 512, overlap 50), all-MiniLM-L6-v2 embedding (384 dimensions) [12], ChromaDB vector database, cosine similarity, and top-K=5. The proposed hybrid system architecture is shown in Figure 1, where BERT filters relevant segments before entering RAG. Evaluation is carried out through an ablation study of five scenarios: S0 (baseline), S1 (RAG), S2 (BERT), S3 (hybrid default), and S4 (hybrid + Bayesian Optimization). The measured metrics are classification performance (F1-macro) and token efficiency (Token Reduction Ratio), with significance tested by the Wilcoxon signed-rank test ($p<0.05$).

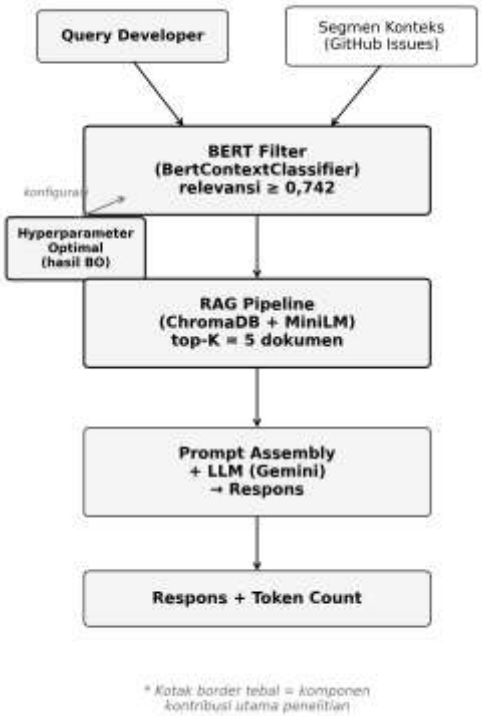


Figure 1. BERT + RAG + Bayesian Optimization Hybrid System Architecture

4. Result And Discussion

Bayesian Optimization was run for 20 trials (8 fully completed, 12 pruned). The best hyperparameters are shown in Table 2, with F1-macro validation of 0.889.

Table 2. Optimal Hyperparameters from Bayesian Optimization Results

Hyperparameter	Optimal Value
Learning Rate	2.04×10^{-5}

Hyperparameter	Optimal Value
Batch Size	32
Epochs	3
Dropout Rate	0.216
Relevance Threshold	0.606
Validation Macro F1-Score	0.889

On the test data (2,419 samples), the model achieved an F1-macro of 0.887 and an accuracy of 0.889 (Table 3). High recall on the relevant class (0.914) is important because the system prioritizes not discarding truly relevant contexts.

Table 3. Classification Results on the Test Set

Class	Precision	Recall	F1-Score	Support
Not Relevant	0.889	0.859	0.873	1,076
Relevant	0.890	0.914	0.902	1,343
Accuracy	—	—	0.889	2,419
Macro Average	0.889	0.886	0.887	2,419

The average token consumption per scenario is shown in Table 4. Only S2 (BERT filter without RAG) saves tokens, amounting to 8.35% compared to the baseline; while S1, S3, and S4 actually increase the number of tokens because RAG adds supporting documents. This finding is consistent with the weakness of pure RAG identified by Lewis [1]—without pre-retrieval filtering, retrieved documents can contain noise. The main finding of this study is that BERT-based relevance filtering effectively suppresses token consumption, while RAG—which adds context—increases it, confirming BERT's role as a token efficiency component.

Table 4. Ablation Study Results and Significance Test

Scenario	Average Tokens	Reduction vs. S0	Wilcoxon *p*-value
S0 Baseline	320.1	0.00%	—
S1 RAG Only	592.6	–85.12%	≈ 0
S2 BERT Only	293.4	+8.35%	1.05×10^{-142}
S3 Hybrid (Default)	551.3	–72.24%	3.72×10^{-168}
S4 Full	573.8	–79.26%	4.40×10^{-159}

The Bayesian Optimization hyperparameter results (F1-validation 0.889) produced better classification performance than the default configuration, demonstrating the benefits of automatic tuning. Pruner dropped 12 out of 20 unpromising trials, demonstrating the sampling efficiency of this method. The findings are in line with Sabbatella [6] who demonstrated Bayesian Optimization as a sample-efficient method for optimization on LLM-based systems. The difference in token consumption between the BERT-only configuration and the configurations involving RAG can be explained by the different functions performed by the two components within the proposed architecture. In the BERT-only scenario (S2), BERT operates as a relevance pre-filter that removes context segments classified as irrelevant before they are processed further. Consequently, only context considered relevant is retained, reducing the amount of information that needs to be passed to the subsequent processing stage. This filtering mechanism directly reduces the input context and explains why S2 achieved an average consumption of 293.4 tokens, compared with 320.1 tokens in the baseline, corresponding to an 8.35% reduction.

In contrast, RAG performs a fundamentally different operation. Rather than removing context, the RAG component retrieves additional documents from the vector database and incorporates them into the prompt supplied to the LLM. In this study, the retrieval process uses 512-token chunks with an overlap of 50 tokens, cosine similarity, and top-K = 5. Therefore, even when the retrieved documents are semantically related to

the query, their inclusion increases the amount of context that must be processed by the LLM. This architectural characteristic explains why the RAG-only scenario (S1) increased the average token consumption to 592.6 tokens compared with 320.1 tokens in the baseline.

The results of the hybrid configurations further demonstrate this trade-off. Although BERT is positioned before RAG and is intended to reduce irrelevant context, the subsequent retrieval mechanism still adds supporting documents to the LLM input. As a result, the filtering benefit provided by BERT does not necessarily translate into lower total token consumption when RAG is activated. The average token consumption reached 551.3 tokens in S3 and 573.8 tokens in S4, both substantially higher than the baseline and the BERT-only configuration. This indicates that the contribution of BERT and RAG should be interpreted according to their distinct architectural roles: BERT is a context-reduction mechanism, whereas RAG is a context-expansion mechanism.

From the perspective of the retrieval mechanism, the observed increase also illustrates a trade-off between context compression and contextual enrichment. BERT attempts to minimize the amount of information entering the downstream process by identifying segments that are relevant to the issue. RAG, on the other hand, attempts to improve the information available to the LLM by retrieving additional supporting content. Consequently, retrieval relevance does not automatically imply token efficiency. Even relevant retrieved documents require additional tokens when they are incorporated into the LLM prompt. In the present architecture, this additional context outweighs the token savings obtained from pre-filtering when RAG is enabled.

The Bayesian Optimization configuration (S4) does not eliminate this architectural effect. Although optimization improved the validation classification performance to an F1-macro of 0.889, S4 still consumed 573.8 tokens on average. This suggests that improving the BERT classifier's hyperparameters and relevance threshold can improve classification quality but does not fundamentally change the token cost introduced by the retrieval stage. Therefore, the findings indicate that token efficiency in the proposed system is primarily associated with the pre-filtering function of BERT, while the RAG component introduces an additional token cost through context retrieval.

These findings provide an important interpretation of the proposed hybrid architecture. The objective of RAG is not inherently to minimize the number of tokens but to provide additional information that may support the generation process. In this study, however, the evaluation focuses on token consumption rather than the quality of the generated LLM response. Therefore, the higher token consumption observed in the RAG-based scenarios should not be interpreted as evidence that RAG is ineffective. Rather, it demonstrates a trade-off between reducing input context and enriching the available context. This distinction is particularly important because the study does not evaluate response quality using metrics such as ROUGE-L or BERTScore, which are explicitly outside the current scope.

Overall, the ablation results demonstrate that the token-efficiency benefit originates primarily from BERT's ability to eliminate unnecessary context before downstream processing, whereas RAG increases token consumption because its retrieval mechanism introduces additional supporting context. The hybrid architecture therefore reveals a fundamental trade-off between context reduction and context enrichment, providing a clearer explanation of why combining the two components does not automatically produce lower token consumption.

5. Conclusion

BERT-based context relevance filtering was shown to reduce token consumption in software issue-solving tasks by 8.35% (significant, Wilcoxon $p<0.05$), with an F1-macro of 0.887. Bayesian optimization improved

classification performance (F1-validation 0.889). A key finding of this study is that token efficiency stems from the BERT pre-filter component, not from RAG, which adds context.

Study limitations: The dataset is limited to English-language Python repositories; the quality of LLM response content (ROUGE-L, BERTScore) is out of scope; some labels were automatically generated and manually validated on only 15% of the sample. Suggestions for further research: testing generalizability to other languages and domains, and integrating LLM response quality evaluation directly.

6. References

[1] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," Adv. Neural Inf. Process. Syst., vol. 33, pp. 9459-9474, 2020.
[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proc. NAACL-HLT, 2019, pp. 4171-4186.
[3] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in Proc. 25th ACM SIGKDD, 2019, pp. 2623-2631.
[4] C. E. Jimenez et al., "SWE-bench: Can language models resolve real-world GitHub issues?," in Proc. Int. Conf. Learn. Represent. (ICLR), 2024.
[5] N. F. Liu et al., "Lost in the middle: How language models use long contexts," Trans. Assoc. Comput. Linguist., vol. 12, pp. 157-173, 2024.
[6] A. Sabbatella, A. Ponti, I. Giordani, A. Candelieri, and F. Archetti, "Prompt optimization in large language models," Mathematics, vol. 12, no. 6, art. 929, 2024.
[7] A. Vaswani et al., "Attention is all you need," Adv. Neural Inf. Process. Syst., vol. 30, pp. 5998-6008, 2017.
[8] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," arXiv:2104.08691, 2021.
[9] E. J. Hu et al., "LoRA: Low-rank adaptation of large language models," arXiv:2106.09685, 2021.
[10] A. Archetti and A. Candelieri, Bayesian Optimization and Data Science. Berlin: Springer, 2019.
[11] M. Balandat et al., "BoTorch: A framework for efficient Monte-Carlo Bayesian optimization," Adv. Neural Inf. Process. Syst., vol. 33, pp. 21524-21538, 2020.
[12] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in Proc. EMNLP-IJCNLP, 2019, pp. 3982-3992.
[13] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," MIS Quarterly, vol. 28, no. 1, pp. 75-105, 2004.
[14] Y. Gao et al., "Retrieval-augmented generation for large language models: A survey," IEEE Trans. Knowl. Data Eng., 2025.
[15] J. Wei et al., "Chain-of-thought prompting elicits reasoning in large language models," Adv. Neural Inf. Process. Syst., vol. 35, pp. 24824-24837, 2022.