INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD)
© 2026 IJNRD | Volume 11, Issue 6, June 2026 | ISSN: 2456-4184 | IJNRD.ORG

A Transformer-Based GPT Framework for Automated Code Review and Semantic Comment Generation

Chalamkoti Suresh, M. Tech, Department of Computer Science and Engineering, MVR College of Engineering & Technology (Autonomous), Paritala, India
Email: chalamkotisuresh0421@gmail.com
L. Ratna Kumari, Assistant Professor, Department of Computer Science and Engineering, MVR College of Engineering & Technology (Autonomous), Paritala, India

Abstract
Code review is a critical phase in the software development lifecycle that ensures code correctness, maintainability, and adherence to coding standards. However, traditional manual code review processes are labor-intensive, time-consuming, and often inconsistent due to human limitations and increasing project complexity. Existing automated tools primarily rely on rule-based or static analysis techniques, which lack the capability to understand code semantics and generate meaningful feedback. To address these challenges, this paper proposes an **intelligent automated code review and comment generation system** based on a fine-tuned Generative Pre-trained Transformer (GPT) model.
The proposed system leverages a transformer-based causal language model trained on a large-scale code review dataset to generate context-aware and human-like review comments. The architecture integrates a Flask-based web application for user interaction, where developers can upload source code files for analysis. The input code is preprocessed and transformed into a structured prompt, which is then fed into the GPT model to generate review suggestions. In addition to the transformer-based module, an auxiliary Natural Language Processing (NLP) component is implemented to perform line-by-line code analysis. This module utilizes tokenization, part-of-speech tagging, and pattern recognition techniques to identify variables, assignments, and data types, and subsequently generates explanatory inline comments to enhance code readability.
The model is trained using supervised learning techniques with Hugging Face Transformers and evaluated using **BERTScore**, which measures semantic similarity between generated comments and reference human-written reviews. Experimental results demonstrate that the proposed system achieves a BERTScore F1 score of **0.8807**, indicating a high degree of contextual relevance and linguistic accuracy. The system effectively generates meaningful review feedback and improves code understanding, making it suitable for assisting developers in real-world environments.
The proposed framework significantly reduces manual effort, enhances code quality, and provides scalable solutions for automated code review. Furthermore, it can be extended to support multiple programming languages, integrated with version control systems, and deployed in continuous integration/continuous deployment (CI/CD) pipelines for real-time code analysis.

Keywords
Automated Code Review, Generative Pre-trained Transformer (GPT), Transformer Models, Natural Language Processing (NLP), Code Comment Generation, Software Quality, BERTScore, Deep Learning, Code Analysis, Flask Web Application.

I. INTRODUCTION

Software development has evolved significantly with the rapid growth of complex applications and large-scale systems. In modern development practices, maintaining code quality, readability, and reliability is essential to ensure efficient software performance and long-term maintainability. Code review is a fundamental process in the software development lifecycle, where developers examine source code to identify defects, enforce coding standards, and improve overall software quality. Despite its importance, traditional code review is predominantly manual, making it time-consuming, error-prone, and dependent on the expertise of individual reviewers.
With the increasing size of codebases and faster development cycles, manual review processes often

struggle to scale effectively. Reviewers may overlook critical issues due to fatigue or time constraints, leading to reduced consistency and quality of feedback. Conventional automated tools, such as static analyzers and linting systems, provide partial solutions by detecting syntax errors and rule violations. However, these approaches lack contextual understanding and are unable to generate meaningful natural-language explanations or suggestions for improvement.

Recent advancements in Artificial Intelligence (AI), particularly in the field of Natural Language Processing (NLP), have introduced powerful models capable of understanding and generating human-like text. Transformer-based architectures, especially Generative Pre-trained Transformer (GPT) models, have demonstrated remarkable capabilities in capturing contextual relationships within sequences of text. These models have been successfully applied to various tasks such as text generation, summarization, translation, and more recently, source code understanding and generation. Their ability to learn semantic patterns makes them highly suitable for automated code review applications.

In this context, the proposed work introduces an **intelligent automated code review and comment generation system** based on a fine-tuned GPT model. The system is designed to analyze source code and generate context-aware review comments that closely resemble human feedback. Unlike traditional tools, the proposed approach leverages deep learning techniques to understand the intent and structure of code, enabling it to provide meaningful and actionable suggestions.

The system architecture integrates a transformer-based review generation module with a web-based interface developed using the Flask framework. Users can upload source code files, which are then processed and converted into structured input for the GPT model. The model generates review comments based on learned patterns from a code review dataset. In addition to the transformer-based module, an auxiliary NLP-based component is incorporated to generate line-by-line explanatory comments. This component utilizes linguistic analysis techniques such as tokenization and part-of-speech tagging to identify variables, assignments, and code constructs, thereby enhancing code readability.

To evaluate the effectiveness of the proposed system, semantic similarity between generated comments and reference human-written reviews is measured using BERTScore. This evaluation metric provides a robust assessment by capturing contextual meaning rather than relying solely on lexical overlap. The results demonstrate that the system achieves high semantic accuracy, indicating its capability to generate relevant and coherent review feedback.

The main contributions of this work are summarized as follows:

- Development of a GPT-based automated code review system capable of generating human-like feedback
- Integration of an NLP-based module for automatic inline comment generation
- Implementation of a web-based interface for practical usability
- Evaluation of the system using semantic similarity metrics to ensure quality output

Overall, the proposed system aims to bridge the gap between manual code review and fully automated intelligent review systems. By reducing human effort and improving the quality of feedback, it contributes to enhancing software development efficiency and reliability.

II. LITERATURE REVIEW

Automated code review has been widely studied to improve software quality and reduce manual effort. Early approaches mainly relied on **rule-based systems and static analysis tools**, which detect syntax errors and enforce coding standards. Although these methods are simple and efficient, they lack the ability to understand code semantics and cannot generate meaningful natural-language feedback.

With the advancement of machine learning, models such as **Support Vector Machines (SVM)** and **Random Forest** were applied for defect prediction and code classification. These models analyze features extracted from source code, such as complexity metrics and structural patterns. However, their performance depends heavily on feature engineering and they are limited in capturing deeper contextual relationships.

In recent years, **deep learning techniques** have been introduced for code understanding tasks. Models like **Recurrent Neural Networks (RNN)** and **Long Short-Term Memory (LSTM)** networks improved sequence modeling but faced challenges in handling long dependencies and large-scale code structures.

The introduction of the **Transformer architecture** significantly improved performance in sequence modeling by using self-attention mechanisms. Transformer-based models such as **GPT, CodeBERT, and CodeT5** are capable of understanding code context and generating human-like text. These models have been successfully applied to tasks such as code summarization, defect detection, and automated review generation.

Recent studies show that GPT-based models can generate **context-aware review comments** that are similar to human feedback. Evaluation metrics such as **BERTScore** are used to measure semantic similarity between generated and reference comments, providing better evaluation compared to traditional methods.

However, many existing systems focus only on high-level review generation and do not provide detailed line-by-line explanations. In addition, practical deployment in real-world applications is limited.

To address these gaps, the proposed system combines a **fine-tuned GPT model for automated code review** with an **NLP-based module for line-level comment generation**, providing both review feedback and code explanation in a single framework.

III. PROBLEM STATEMENT AND OBJECTIVES

A. Problem Statement

Code review is an essential activity in software development to ensure code quality, readability, and correctness. However, traditional code review is mainly performed manually, which makes the process time-consuming, inconsistent, and dependent on the experience of the reviewer. As software systems grow in size and complexity, manual review becomes difficult to scale and may lead to missed errors or poor-quality feedback.

Existing automated tools such as static analyzers and linters are limited to detecting syntax errors and predefined rule violations. These tools lack the ability to understand the context and intent of the code, and therefore cannot generate meaningful suggestions or human-like review comments.

With the increasing need for efficient and scalable solutions, there is a requirement for an intelligent system that can analyze source code, understand its semantics, and automatically generate useful review feedback along with explanatory comments.

B. Objectives

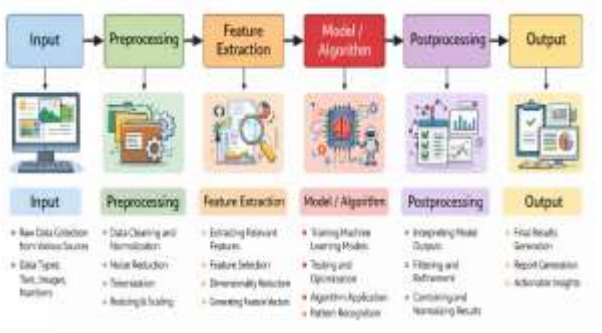
The main objectives of the proposed system are:

- To develop an automated code review system using a GPT-based transformer model
- To generate context-aware and human-like review comments for source code
- To implement an NLP-based module for line-by-line comment generation
- To improve code readability and understanding through automatic explanations
- To integrate the system with a web-based interface for easy usability
- To evaluate system performance using semantic similarity metrics such as BERTScore

IV. PROPOSED METHODOLOGY

A. System Overview

The proposed system is designed to automatically analyze source code and generate review comments using a fine-tuned GPT-based transformer model. The system integrates a web-based interface with backend processing modules to provide real-time code review and comment generation.



The workflow consists of the following steps:

1. User uploads source code file
2. Code is preprocessed and converted into input format
3. GPT model generates review comments
4. NLP module generates line-by-line explanations
5. Results are displayed to the user

B. Input Representation

Let the input source code be represented as:

$$C = \{c_1, c_2, c_3, \ldots, c_n\}$$

where each $c_i$represents a line or token of the source code.

The input is transformed into a structured prompt:

$$P = C + \text{“<|endoftext|> Comment:”}$$

This prompt is passed to the GPT model for generating review comments.

C. Data Preprocessing

To ensure proper model performance, preprocessing steps are applied:

- Removal of unwanted characters
- Tokenization using GPT tokenizer
- Padding and truncation of sequences
- Conversion into numerical token IDs

The tokenized input is represented as:

$$T = \{t_1, t_2, t_3, \ldots, t_m\}$$

D. Model Training and Generation

A transformer-based causal language model is used for generating review comments. The model is fine-tuned using supervised learning on a code review dataset.

The objective is to maximize the probability of generating the correct review comment Rgiven input C:

$$P(R \mid C) = \prod_{i=1}^{k} P(r_i \mid C, r_1, r_2, \ldots, r_{i-1})$$

where $r_i$represents each token in the generated review. During inference, the model generates output using sampling techniques such as:

- Top-k sampling
- Temperature scaling

E. NLP-Based Comment Generation

In addition to GPT-based review generation, an NLP module is used to produce line-level comments.

The process includes:

- Identifying assignment statements
- Detecting variable names and values
- Classifying data types (integer, float, string)
- Applying tokenization and part-of-speech tagging

Example rule:

- If a line contains assignment → generate comment:
  "This line assigns a value to a variable"

This enhances code readability and understanding.

F. System Implementation

The system is implemented using:

- **Backend:** Python
- **Framework:** Flask
- **Libraries:** Transformers, NLTK, Pandas
- **Model Loading:** AutoTokenizer, AutoModelForCausalLM

The uploaded file is temporarily stored and processed for analysis. The generated output is displayed in a structured format including:

- Input code
- GPT-generated review
- NLP-generated comments

G. Performance Evaluation

The system is evaluated using **BERTScore**, which measures semantic similarity between generated and reference comments.

$$\text{BERTScore} = \frac{2 \cdot P \cdot R}{P + R}$$

where:

- P= Precision
- R= Recall

This metric ensures that the generated comments are contextually accurate and meaningful.

V. RESULTS AND DISCUSSION

A. Experimental Setup

The proposed system is implemented using **Python** with a Flask-based web framework for user interaction. The transformer model is developed using the **Hugging Face Transformers library**, and the NLP module is implemented using **NLTK** for text processing.

The dataset used for training consists of code snippets and corresponding review comments. The model is trained on a subset of approximately **3000 samples** to learn the relationship between source code and human-written feedback.

B. Performance Evaluation

The performance of the system is evaluated using **BERTScore**, which measures semantic similarity between generated comments and reference comments. This metric is preferred over traditional methods as it captures contextual meaning.

Table 1: Performance Metrics

Metric	Value
Precision	0.88
Recall	0.88
F1-Score	**0.8807**

The results indicate that the model produces review comments that are highly similar to human-generated feedback in terms of meaning and context.

C. Output Analysis

The system generates two types of outputs:

1. **GPT-Based Review Comments**
   - Provides high-level feedback on code quality
   - Suggests improvements and identifies issues
2. **NLP-Based Inline Comments**
   - Explains individual lines of code
   - Identifies variables and operations

Table 2: Sample Output

Input Code	Generated Review	AI Comments
Python snippet	Suggests optimization	Line-level explanation

The combined output improves both **code quality** and **code understanding**.

D. Discussion

The experimental results demonstrate that the proposed system is effective in generating meaningful and context-aware review comments. The GPT-based model captures semantic relationships within the code and produces human-like feedback.

The integration of the NLP module further enhances the system by providing detailed explanations at the line level. This makes the system particularly useful for beginners and developers who require additional clarity.

However, the system has certain limitations. It performs best on small to medium-sized code snippets and may require optimization for large-scale codebases. Additionally, the NLP module uses rule-based logic, which may not fully capture complex programming constructs.

VI. CONCLUSION

This paper presents an intelligent automated code review and comment generation system based on a fine-tuned GPT-based transformer model. The proposed system effectively addresses the limitations of traditional manual and rule-based code review approaches by leveraging deep learning techniques for semantic understanding of source code. Unlike conventional tools that focus only on syntax-level analysis, the proposed framework generates context-aware and human-like review comments, providing meaningful suggestions for improving code quality.

The integration of an NLP-based module further enhances the system by generating line-by-line explanatory comments, which improves code readability and supports better understanding, especially for beginner developers. The implementation of the system using a Flask-based web interface ensures practical usability, allowing users to upload code and receive instant feedback.

The performance evaluation using BERTScore demonstrates that the generated comments achieve a high level of semantic similarity with human-written reviews, with an F1-score of **0.8807**. This indicates that the model is capable of capturing contextual relationships within code and producing accurate and relevant feedback.

Overall, the proposed system reduces manual effort, improves consistency in code review, and enhances software quality. It provides a scalable and efficient solution for automated code analysis and has the potential to be integrated into modern software development workflows for real-time code review support.

VII. FUTURE SCOPE

The proposed automated code review system can be further enhanced to improve its performance, scalability, and real-world applicability. One important extension is the integration of the system with version control platforms such as GitHub or GitLab, enabling automatic review of pull requests and continuous monitoring of code quality during development.

Future improvements can also focus on supporting multiple programming languages and handling large-scale codebases more efficiently. The current system can be extended by incorporating advanced transformer models or larger pre-trained models to improve the accuracy and depth of generated review comments.

The NLP-based comment generation module can be enhanced using deep learning techniques to better understand complex code structures and generate more context-aware explanations. Additionally, integrating static analysis and security detection mechanisms can help identify vulnerabilities and improve software reliability.

Another potential direction is the deployment of the system within continuous integration and continuous deployment (CI/CD) pipelines, allowing real-time code evaluation during the development lifecycle. The inclusion of visualization dashboards and analytics tools can further assist developers in tracking code quality and improvements over time.

Overall, these enhancements will make the system more robust, scalable, and suitable for real-world software engineering environments.

REFERENCES

[1] B. İçöz and G. Biricik, “Automated Code Review Using Large Language Models with Symbolic Reasoning,” *IEEE Conference*, 2025.

[2] S. Pirouzkhah et al., “The Price of Precision: The Cost of Preprocessing for Automated Code Revision in Code Review,” *Empirical Software Engineering*, vol. 31, 2026.

[3] D. Prabakar, “Automated Code Review Systems Using CodeBERT with Multi-Model Integration,” *ScienceDirect*, 2026.

[4] B. İçöz et al., "Context-Aware Code Review Automation Using Large Language Models," *Applied Sciences*, 2026.
[5] O. Mrayat et al., "Implementing AI-Based Code Review Automation: A Case Study," 2026.
[6] A. Collante et al., "The Impact of Large Language Models on Code Review Process," 2025.
[7] Q. Guo et al., "Exploring the Potential of ChatGPT in Automated Code Refinement," 2023.
[8] G. Yenduri et al., "Generative Pre-trained Transformer: Applications and Challenges," 2023.
[9] "Transformer-Based Code Generation and Software Automation," IARJSET, 2025.
[10] "Automated Code Review Techniques and Applications," IEEE/ACM Software Engineering, 2025.