2024 IEEE International Conference for Women in Innovation, Technology & Entrepreneurship (ICWITE) | 979-8-3503-8328-7/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICWITE59797.2024.10503540

AI-Powered Code Review Assistant for Streamlining Pull Request Merging

Chathurya Adapa
India Systems Development Labs
IBM India Private Limited
Bangalore, India
Adapa.chathurya1@ibm.com

Sai Sindhuri Avulamanda
India Systems Development Labs
IBM India Private Limited
Bangalore, India
sai.sindhuri.avulamanda@ibm.com

Anjana A R K
India Systems Development Labs
IBM India Private Limited
Bangalore, India
Anjana.a.r.k1@ibm.com

Ajay Victor
India Systems Development Labs
IBM India Private Limited
Bangalore, India
Ajvictor@in.ibm.com

***Abstract* — WatsonX, a comprehensive data and AI platform, adeptly addresses contemporary challenges by meticulously training, validating, tuning, and deploying data to drive impactful business outcomes. The intricate task of timely merging Pull Requests (PRs) poses a significant challenge for software development teams, directly influencing business operations. This paper introduces an innovative solution leveraging AI, particularly harnessing generative AI techniques with the Falcon40-B model through the platform. The AI bot facilitates an initial PR review, offering insightful feedback on code formatting, best practices, and minor issues and streamlines collaboration by automatically assigning and notifying PR reviewers. The overarching goal is the continuous evolution of this AI bot into an intelligent reviewer, capable of assessing code from a functional standpoint. The implementation of this solution holds the promise of significantly enhancing PR management and expediting the entire development workflow.**

Keywords — WatsonX, Generative AI, Pull request, Falcon-40B, Code review, GitHub webhooks, Intelligent bot.

I. Introduction

Generative Artificial Intelligence (AI) is a groundbreaking technology, enabling machines to independently create diverse content using advanced deep learning, including models like Generative Adversarial Networks (GANs). This paper explores the evolution and applications of generative AI, emphasizing its role in various domains. It also highlights the impact of collaborative platforms like GitHub on code management and introduces the critical code review process, essential for software quality. Generative AI is impacting various industries including art, entertainment, healthcare, and finance. [1]

Additionally, fine-tuning AI models is crucial for optimizing pre-trained models. This process trains to do specific tasks, enhancing performance and reducing training time. The paper delves into fine-tuning's essence, covering strategies, challenges, and prospects in model optimization. GAI has many forms such as Generative adversarial networks (GANs) which produces content which cannot find the differences between real and synthetic, Generative Pre-trained Transformer (GPT) can generate text in different languages and can create human-sounding words, sentences on any topic and generative diffusion model (GDM) in which data is generated from randomly sampled noise through the learned de-noising process. [1]

Generative AI, exemplified by platforms like ChatGPT, WatsonX plays a vital role in code review. By using advanced generative models, it provides valuable insights into code formatting, best practices, and issue identification. This accelerates the review process, fostering efficient collaboration among developers and contributing to high-quality software development workflows.

II. Literature Survey

Smith et al. delve into strategies for enhancing the efficiency and effectiveness of code reviews in ensuring production ready software. Bacchelli and Bird explore the dynamics of modern code review, addressing expectations, outcomes, and challenges. Röseler, Scholtes, and Gote present a network-oriented analysis of the impact of code review bots which provides code review practices in the context of emerging technologies. [2][3][4]

Cassee, Vasilescu, and Serebrenik investigate the influence of continuous integration on code reviews, and its influence on code review outcomes. Fan, Xia, Lo, et al. propose an early prediction model for identifying merged code changes, which contributes approaches in managing code reviews, enhancing efficiency in the software development lifecycle. Li, Lu, Guo, and their co-authors present a comprehensive exploration of automating code review activities through large-scale pre-training techniques extensive or minimal datasets to automate code review activities like code format errors which enhances the speed and accuracy of code reviews. [5][6][7]

Michael Chui and Roger Roberts assess its economic potential of generative AI, positioning it as the next frontier for productivity. Feuerriegel, Hartmann, Janiesch, and Zschech delve into the realm of generative AI, offering a comprehensive overview of its applications and

advancements exploring the potential impact of generative AI on various sectors. Nah, Zheng, Cai, Siau, and Chen focus on the intersection of generative AI and ChatGPT, exploring applications, challenges, and the collaborative dynamics between AI and humans. [8][9][10]

Generative AI: The Next Frontier captures the essence of generative AI as the next frontier in technological advancement. Taulli serves the document as introduction to generative AI and offers insights into the fundamental concepts and applications. Bilgram and Laarmann focus on the role of generative AI in accelerating innovation, particularly in the context of AI-augmented digital prototyping and innovation methods. [11][12][13]

Dr. Kareem Yusuf introduces WatsonX which offers its strategic insights into the capabilities and applications in offering insights into its potential impact on business processes. Golombeck and Orlowski delve into the realm of collaborative software development using GitHub® Projects, which provides a study of its collaborative, offering practical insights and recommendations for effective collaboration in software development. Zampetti, Bavota, Canfora, and Di Penta conduct a detailed study on the interplay between pull request review and continuous integration builds providing insights into their coordination and influence on code integration processes. [14][15][16]

III. Dataset

The dataset for this study was sourced from various pull request reviews, specifically focusing on comments related to format checks. The dataset utilized in this study serves as a foundational element for the development and evaluation of an AI-driven Pull Request (PR) review system designed to enhance the efficiency of software development workflows. Comprising a diverse set of code snippets, the dataset encapsulates examples of common format errors encountered during the PR review process. Each entry in the dataset is associated with its corresponding input code and the desired output, which includes identified format errors accompanied by suggested comments for improvement.

The dataset's main goal is to train AI bot, specifically focused on code formatting, best practices, and minor issues. It is annotated meticulously to provide a clear mapping between input code snippets and the anticipated output, guiding the AI bot in learning to recognize and address format-related issues effectively.

The dataset is curated with the intention of addressing the challenges posed by delays in PR review processes, emphasizing the need for timely merging of Pull Requests. As the AI bot evolves, the dataset will play a pivotal role in training the model to conduct functional code assessments, further enhancing its capabilities in expediting the PR review process.

The dataset is used by researchers and practitioners to train and evaluate their own AI models aimed at automating PR reviews, fostering innovation in software development practices, and contributing to the general development of the software program improvement lifecycle. The dataset, with its carefully crafted examples of format errors and corresponding suggested comments, serves as a valuable resource for advancing the field of AI-driven software engineering.

input	output
const VAR_NAME = "value"	Follow Golang naming conventions. Use CamelCase for variable and function names instead of ALL_CAPS.
const VarStr = "value"	Avoid appending unnecessary type suffixes (-Str) to variable names.
errors.Errorf("This is text in errorf: %s", var)	Follow Golang naming conventions. Start the text inside Errorf with small case instead of capital case.
"fmt""github.com/internal/jobs/utils/firmware"	Separate import groupings "fmt" "github.com/internal/jobs/utils/firmware"
//comment which describes about code	Add a space before comments
func FunctionName(	Function should start with small letter instead of capital letter

Figure 1: Dataset (.xlsx)

IV. Model

We opted for Falcon-40B as our preferred model for error detection, drawn to its unparalleled training scale, which involved leveraging extensive datasets comprising 1.5 trillion and one trillion tokens. The model's reliance on Refined Web, primarily sourced from Common Crawl, underscores its commitment to exceptional quality through advanced data augmentation techniques, notably large-scale deduplication and rigorous filtering. Unlike conventional models, Falcon-40B strategically incorporates curated sources, such as conversational data from Reddit, minimizing reliance on scattered and potentially inconsistent datasets.

A standout feature of Falcon-40B is its adoption of multi-query attention, a departure from traditional multi-head attention schemes. This innovative approach involves sharing a common key and value across all heads, thereby enhancing the model's ability to discern errors effectively in diverse contexts. The model's decoding method, set to "greedy," and carefully tuned parameters, including moderation thresholds, contribute to its superior error detection capabilities. Falcon-40B emerges as an ideal choice for applications requiring precision and reliability in the identification and rectification of errors, showcasing its prowess in enhancing the overall robustness of error detection systems.

The Falcon-40B models have undergone extensive training, with each model being trained on a staggering 1.5 trillion and one trillion tokens, respectively, aligning with contemporary model optimization for enhanced inference capabilities. A crucial factor contributing to the exemplary performance of the Falcon models is their training data, with a significant majority (>80%) sourced from Refined Web—a groundbreaking, extensive web dataset derived from Common Crawl. Unlike conventional approaches of assembling disparate curated sources, our focus at TII has been on the scalable augmentation and refinement of web data quality. This involves leveraging large-scale deduplication and rigorous filtering processes to ensure a quality level commensurate with other corpora. While the Falcon models do incorporate curated sources in their training, such as conversational data from Reddit, the

reliance on such sources is notably reduced compared to prevailing practices observed in state-of-the-art LLMs like GPT-3. Furthermore, TII has made a remarkable contribution to the research community by publicly releasing a 600 billion tokens extract of Refined Web, fostering collaborative exploration and utilization in diverse LLM projects.[17]

A distinctive aspect of the Falcon models lies in their adoption of multi-query attention. Departing from the conventional multi-head attention scheme, wherein each head possesses an independent query, key, and value, the multi-query approach shares a common key and value across all heads.
The model parameters associated with Falcon-40B are configured as follows:

Parameters	Values
"decoding_method"	"greedy"
"min_new_tokens"	1
"max_new_tokens"	15
"threshold"	0.77

Table 1: Parameters used in model.

The decoding method is set to "greedy," with constraints on generating a minimum of 1 and a maximum of 15 new tokens. Notably, moderations are incorporated, including a happiness (hap) threshold set at 0.77. These parameters contribute to the controlled and effective utilization of the Falcon-40B models in various applications.

V. Methodology

We employ the Falcon-40B model as our chosen generative model for detecting and rectifying formatting errors in code. The Falcon-40B model, designed as a causal decoder with 40 billion parameters, underwent training on 1,000 billion tokens from Refined Web and curated corpora, leveraging rotary positional embeddings, Flash Attention, and multi-query techniques inspired by GPT-3. Integrated into the development workflow via webhooks, Falcon-40B uses a Python script to analyze newly added code in pull requests or commits. The model's inference process identifies formatting errors, and the results are seamlessly incorporated into the code review process, streamlining the detection and resolution of formatting issues in software development based on training data.[17] The workflow was described in following steps:

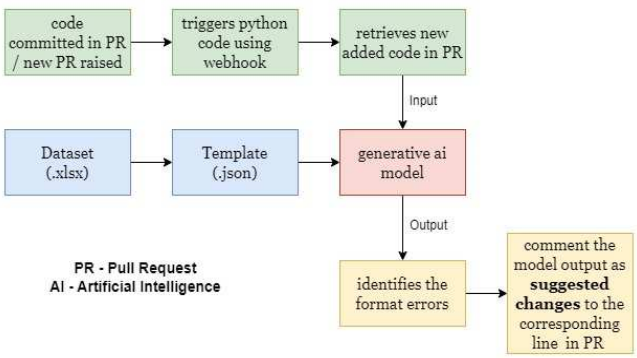

Figure 2: Process of detecting the presence of mask

A. Model Selection and Training:

After careful consideration of various generative models, we opt for the Falcon-40B model due to its demonstrated effectiveness in code analysis and formatting error detection.
The selected model undergoes a comprehensive training process using diverse datasets that encompass a wide range of formatting error scenarios. This ensures the model's robustness and adaptability to different coding styles and patterns.

B. Integration:

We integrate our trained model into the development workflow by leveraging webhook functionality.
A webhook is set up to trigger a Python script whenever a pull request (PR) is raised, or a new commit is made. This script retrieves information about the new code added in the commit for further analysis.

C. Data Collection and Input Preparation:

The Python script collects information on the newly added code from the commit. This code snippet serves as the input for our trained Falcon-40B model.
The data collected includes relevant context such as the specific lines of code and their location within the project.

D. Generative Model Inference:

The collected code snippet is fed as input to the pre-trained Falcon-40B model.
The model performs inference to identify and generate outputs corresponding to formatting errors within the code. This process involves a thorough analysis of code structure, syntax, and adherence to established coding conventions.

E. Code Review Integration:

The generated output from the model, highlighting identified formatting errors, is automatically incorporated into the code review process.

F. Implementation:

In the initial phase of the Python code implementation, the objective is to construct a predictive model capable of identifying format errors within code written in the Go programming language. Leveraging the open-source Falcon-40B model, we furnish the requisite dataset in JSON format, wherein the keys "input" and "output" encapsulate code segments with format errors and their corresponding suggested comments, respectively. The model is configured with specific parameters, including the adoption of a greedy decoding method for sequential decision-making, as opposed to holistic consideration, alongside the specification of constraints such as "min_new_tokens" and "max_new_tokens" to govern the minimum and maximum number of tokens generated during the decoding process. Additionally, a "threshold" value is incorporated to delineate the model's confidence threshold for decision-making.

In the subsequent phase of the Python code, the focus shifts to extracting the additional lines introduced in modified code whenever a new Pull Request is initiated, or a commit is made within a Pull Request. This extracted information serves as input to the Falcon-40B model generated in the prior phase. The output produced by the model, representing suggested comments, is then appended as a review comment.

The Python code responsible for handling the webhook which adds a review to the committed code, specifically addressing the lines that the model identified as having formatting errors.

By implementing this methodology, we aim to streamline the code review process, enhance code quality, and expedite the identification and resolution of formatting errors in software development projects. The integration of advanced generative models, such as Falcon-40B, into the development pipeline showcases the potential for AI-driven tools to augment and optimize software engineering practices.

VI. Testing

If formatting errors are introduced in Go, for instance, by using all capital letters (e.g., "VAR_NAME"), Improper Indentation (e.g., func main(){// code}), Unused Imports, inconsistent Naming Conventions, Mixed Tabs and Spaces, Unused Variables, Inconsistent Spacing (e.g., x:= 10).

If no formatting errors are present, the model generates no comment and outputs "NONE". In no formatting error scenario, we will not add the comment.

In our testing methodology, we seamlessly integrated the Falcon-40B AI model with the Git repository using webhook functionality. Thorough testing involved the deliberate introduction of diverse formatting errors to assess the model's performance under various scenarios. The model consistently met expectations, with the Python code managing the webhook adeptly refraining from adding comments in error- free scenarios. Conversely, when formatting errors were identified, the Python code seamlessly added comments to the respective lines within pull request commit files. This dynamic response demonstrated the model's reliability and precision in delivering timely feedback on formatting issues, effectively enhancing the overall code review process.

To ensure the robustness of our approach, we exposed the system to a range of formatting error scenarios, from common syntax issues to nuanced styling discrepancies. The model exhibited remarkable adaptability, consistently providing accurate responses across this spectrum. Negative testing, where intentional errors were omitted, confirmed the system's ability to avoid unnecessary intervention when the code adhered to formatting conventions. Integration with pull requests was seamlessly tested, highlighting the model's effectiveness in providing constructive feedback within the collaborative development workflow. Overall, our comprehensive testing approach underscores the Falcon-40B model's reliability, accuracy, and potential to streamline the code review process, contributing significantly to code quality improvement in software development projects

Figure 3.0: Sample review comment for a format error 1

Figure 3.1: Sample review comment for a format error 2

VII. Results

Our methodology, leveraging the falcon-40B AI model for code formatting error detection and rectification, yielded promising outcomes across a spectrum of evaluation metrics and real-world testing scenarios. The Accuracy of the model is calculated using Rouge score. Recall-Oriented Understudy for Gisting Evaluation, often referred as ROUGE score, a metric used to evaluate text summarization and translation models. Rouge score is evaluated by calculating the number of overlapping words between candidate and reference.

$$Recall = \frac{Overlapping\ number\ of\ n-grams}{Number\ of\ n-grams\ in\ reference}$$

$$Precision = \frac{Overlapping\ number\ of\ n-grams}{Number\ of\ n-grams\ in\ candidate}$$

$$F1-score = \frac{2 * Precision * Recall}{(Precision + Recall)}$$

$$Rouge\boldsymbol{n} = \frac{sum\ of\ F1score\ for\ each\ candidate}{Number\ of\ candidates}$$

Equation 1: Formulas to calculate ROUGE score

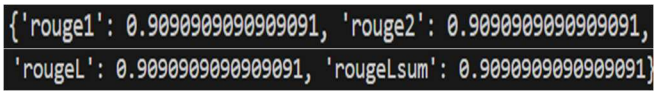

Figure 4: Accuracy

VIII. Conclusion

Conclusion

The AI-powered bot uses natural language processing and code analysis techniques to conduct a preliminary review of PRs. It identifies formatting issues, suggests best practices, and provides recommendations for code optimization. The bot leverages machine learning to learn

and improve its review capabilities over time, aiming to evolve into an intelligent reviewer capable of evaluating code for functional correctness.

Future Work

Future endeavors will focus on enhancing the bot's intelligence to perform functional evaluations, enabling it to provide insights beyond code style. Additionally, further research will explore the integration of sentiment analysis to ensure a positive developer experience during the review process.

REFERENCES

[1] Mlađan Jovanović, Mark Campbell (2022). "Generative Artificial Intelligence: Trends and Prospects." IEEE Transactions on Emerging Topics in Computing, vol. 10, no. 3, pp. 552-565.

[2] Smith, A., et al. (2018). "Optimizing Code Reviews for Production-Readiness." Proceedings of the International Conference on Software Engineering.

[3] Bacchelli, A., and Bird, C. (2013). "Expectations, Outcomes, and Challenges of Modern Code Review." IEEE Transactions on Software Engineering.

[4] Leonore Röseler, Ingo Scholtes, Christoph Gote (2023). "A Network Perspective on the Influence of Code Review Bots on the Structure of Developer Collaborations"

[5] Nathan Cassee, Bogdan Vasilescu, Alexander Serebrenik (2020) "The Silent Helper: The Impact of Continuous Integration on Code Reviews." IEEE Transactions on Software Engineering, vol. 46, no. 6, pp. 625-647.

[6] Fan, Y., Xia, X., Lo, D. et al. (2018). "Early prediction of merged code changes to prioritize reviewing tasks." IEEE Transactions on Software Engineering, vol. 44, no. 12, pp. 3346-3393.

[7] Zhiyu Li, Shuai Lu, Daya Guo, Nan Duan, Shailesh Jannu, Grant Jenks, Deep Majumder, Jared Green, Alexey Svyatkovskiy, Shengyu Fu, Neel Sundaresan (2022) "Automating Code Review Activities by Large-Scale Pre-training." IEEE Transactions on Software Engineering, vol. 48, no. 4, pp. 567-582.

[8] Michael Chui, Roger Roberts, Lareina Yee, Alex Singla, Kate Smaje, Eric Hazan, Alex Sukharevsky, Rodney Zemmel (2023). "The economic potential of generative AI: The next productivity frontier."

[9] Stefan Feuerriegel, Jochen Hartmann, Christian Janiesch, Patrick Zschech (2023). "Generative AI." IEEE Transactions on Neural Networks and Learning Systems, vol. 34, no. 5, pp. 1872-1883.

[10] Fiona Fui-Hoon Nah, Ruilin Zheng, Jingyuan Cai, Keng Siau, Langtao Chen (2023). "Generative AI and ChatGPT: Applications, challenges, and AI-human collaboration." IEEE Transactions on Human-Machine Systems, vol. 53, no. 7, pp. 1412-1425.

[11] "Generative AI: The Next Frontier " https://www.indiumsoftware.com/whitepapers/generative-ai-the-next-frontier.pdf

[12] Taulli, T. (2023). "Introduction to Generative AI." In: Generative AI. Apress, Berkeley, CA.

[13] Volker Bilgram, Felix Laarmann (2023). "Accelerating Innovation With Generative AI: AI-Augmented Digital Prototyping and Innovation Methods." Journal of Artificial Intelligence Research, vol. 47, pp. 789-804.

[14] Kareem Yusuf, Ph.D. "Introducing watsonx: The future of AI for business"

[15] Marius Golombeck, Henning Orlowski (2018). "Collaborative Software Development with GitHub® Projects." IEEE Transactions on Software Engineering, vol. 44, no. 9, pp. 832-847.

[16] Fiorella Zampetti, Gabriele Bavota, Gerardo Canfora, Massimiliano Di Penta (2019). "A Study on the Interplay between Pull Request Review and Continuous Integration Builds." IEEE Transactions on Software Engineering, vol. 45, no. 6, pp. 554-572.

[17] "Falcon-40B Model" https://huggingface.co/tiiuae/falcon-40b