IJIRSET
International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

e-ISSN: 2319-8753 | p-ISSN: 2347-6710

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 13, Issue 6, June 2024

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.423

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.423

|www.ijirset.com | A Monthly Peer Reviewed & Referred Journal | e-ISSN: 2319-8753; p-ISSN: 2347-6710|

|Volume 13, Issue 6, June 2024|

|DOI: 10.15680/IJIRSET.2024.1306164|

From Code to Context: Leveraging Large Language Models for Summarization

Anchit Mhatre, Prof. Rashmi Gourkar

Mumbai Educational Trust's Institute of Computer Science, Mumbai, Maharashtra, India

Mumbai Educational Trust's Institute of Computer Science, Mumbai, Maharashtra, India

ABSTRACT: In software engineering, code summarising is an essential process that aims to automatically produce precise and succinct descriptions of source code. The performance of code summarization jobs has increased dramatically due to recent developments in large language models (LLMs). This paper offers a thorough analysis of the state-of-the-art in code summarization with LLMs, emphasising the approaches, outcomes, and difficulties in this area. We talk about how LLMs can be used for code summarising in contextual code switching, zero-shot learning, and few-shot training. We investigate the use of knowledge bases to direct model testing, automatic semantic augmentation of prompts, and code explanation using LLMs. The paper's conclusion highlights the need for more research by describing the potential paths and unresolved issues in code summarization using LLMs.

KEYWORDS: Code Summarization, Large Language Models, Software Engineering, Natural Language Processing, Artificial Intelligence, Code Analysis, Automatic Description Generation, Contextual Code Switching, Zero-shot Learning, Few-shot Training, Knowledge Bases, Semantic Augmentation, Code Explanation

I. INTRODUCTION

Large-scale software systems are becoming harder for developers to understand and maintain due to the explosion in source code amount and complexity brought about by the rapid rise of software development. Code summary is an essential component of software development that entails producing succinct and educational functionalities of the code summarised. For a number of reasons, such as code reuse, bug finding, and developer onboarding, this activity is crucial. Conventional methods of code summarization focus on human processes like examining source code structures and interpreting comments in the code. These techniques take a lot of time, require a lot of work, and are prone to mistakes. Large language models (LLMs) offer a new paradigm for autonomous code summarization, which has the potential to completely transform the field of code summarization.

Natural language processing tasks including text categorization, sentiment analysis, and machine translation have been remarkably successful for LLMs. LLMs have recently been applied to software engineering tasks, like as code summarization, by researchers. The goal is to make use of LLMs' ability to analyse source code and produce precise and succinct summaries.

The state-of-the-art research on autonomous code summarization using LLMs is thoroughly reviewed in this work. We will discuss current developments in autonomous semantic augmentation for code summarization, few-shot learning, and transfer learning. We will also talk about the difficulties LLMs encounter while summarising code, such as the labor-intensive nature of binary code summarization, managing lengthy context scenarios, and bridging the semantic divide between code and normal language.

II. LITERATURE REVIEW

Recent years have seen a substantial increase in interest in autonomous code summarization utilising Large Language Models (LLMs) because of its potential to decrease code comprehension difficulty and increase software development efficiency. With a focus on important discoveries, difficulties, and potential paths forward, this review of the literature seeks to give a broad picture of the state of the discipline today.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.423

|www.ijirset.com | A Monthly Peer Reviewed & Referred Journal | e-ISSN: 2319-8753; p-ISSN: 2347-6710|

|Volume 13, Issue 6, June 2024|

|DOI: 10.15680/IJIRSET.2024.1306164|

III. BACKGROUND

Software engineers find automatic code summarising to be an essential duty as it facilitates their understanding of the functionality and intent of code in a timely manner. Conventional methods for summarising code depend on domain-specific expertise and manual annotation. These techniques, however, take a lot of time and frequently call for a high level of knowledge. Because of LLMs' capacity to comprehend and produce natural language text, new avenues for automatic code summarization have been explored.

IV. KEY FINDINGS

1. Transfer Learning: It has been demonstrated that transfer learning works well to raise LLMs' performance on code summarising tasks. Researchers have significantly improved the summarization quality by training the models on huge natural language datasets and fine-tuning them on code datasets.[1]

By training the LLM on large selection of natural language texts and transferring the model on code datasets and fine tuning that model has turned to be effective.

By implementing a context-based transfer learning for low resource code summarization (LCRS), which is achieved by learning the common information from languages with rich resources and transferring to target language model for further finetuning [2]

2. Few Shot Learning: The application of few shot learning to code summarization has also been studied. This method entails using a smaller dataset for model training and a bigger one for model fine-tuning. Few shot learning has been shown to be useful in helping LLMs perform better on code summary tasks.[3]

Big LLMs like GPT3 and Codex have achieved state of the art performance on several natural language tasks and code. The particular aspect of these LLM are few shot and zero shot learning. The challenge in few-shot learning is the balance between underfitting due to limited supervised samples and overfitting caused by memorizing task-specific features. [4]

3. Automatic Semantic Augmentation: An approach to enhance the performance of code summarization is the automatic semantic augmentation of LLM prompts. Using this method, the LLM prompts are enhanced with semantic facts to aid in the models' comprehension of the code context. Studies have indicated that this method can enhance the precision and caliber of summaries.[3]

The automatic augmentation of LLM prompt with semantic facts can be used explicitly to improve code summarization performance. It can yield a substantial enhancement to LLMs' line completion performance.[3]

4. Transformer-Based Models: Because transformer-based models may capture contextual information and long-range dependencies, they are frequently employed in code summarizing tasks. Researchers have discovered that these models can perform better in terms of accuracy and quality of summarization than conventional methods.[1]

V. CHALLENGES

1. Limited Data: The scarcity of high-quality training data is one of the main obstacles to autonomous code summarization. Numerous scholars have put forth a number of strategies to deal with this problem, such as transfer learning and data augmentation.[1]
2. Code Complexity: Another major obstacle to autonomous code summarization is code complexity. Several approaches have been put forth by researchers to deal with this problem, like application of graph-based models and attention mechanisms.[3]
3. Metrics for Evaluation: The measures used to evaluate autonomous code summarization are still being developed. A number of metrics, such as BLEU and ROUGE, have been developed by researchers to evaluate the quality of code summaries. These measurements do have certain drawbacks, though, and more study is required to provide more useful assessment techniques.[5]

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.423

|www.ijirset.com | A Monthly Peer Reviewed & Referred Journal | e-ISSN: 2319-8753; p-ISSN: 2347-6710|

|Volume 13, Issue 6, June 2024|

|DOI: 10.15680/IJIRSET.2024.1306164|

4. Labor-Intensive Nature of Binary Code Summarization: Binary summarization is challenging for even modern LLMs like GPT-4 to comprehend due to intricate software details and need to grasp it's functionality making it a time and resource intensive task[7]
5. Long Context Scenarios: A context window means the number of tokens the model can consider while processing text. Processing long context requires a high amount of GPU memory which can cost higher, have longer response latency, and subpar performance as compared to short context window. To address this, approaches like prompt compression can be used to improve LLM data perception thus improving performance. [8]
6. Semantic Gap between Natural Language and Code: There is a lot of difference between natural language and code. LLMs need to bridge this gap. While some performance is shown in area of code gen and summarization, autonomous code summarization is still in active research phase. For example, ChatGPT performs significantly worse than several other state of the art models when trained on widely used python dataset.[9]
7. Limited Project-Specific Data for Training: When a code project starts,limited to no history is present of the project. Since there is less information present, the context window is small, leading to inferior output quality. But with advances in research, LLMs like GPT Codex has shown potential to work with limited information and surpasses state of the art models for code summarization.[4]

VI. FUTURE DIRECTIONS

1. Multimodal Learning: Multimodal learning can enhance autonomous code summarization by fusing natural language processing with various modalities like vision and audio. To increase the accuracy of code summarization, researchers have suggested a number of multimodal learning strategies, such as the inclusion of visual and auditory cues.[6]
2. Explainability: One essential component of autonomous code summarization is explainability. Researchers have suggested a number of techniques, such as the use of attention mechanisms and visualisations, to increase the explainability of code summaries.[3]
3. Integration with Development Tools: Another important field of study is integration with development tools. Numerous techniques, such as the usage of APIs and plugins, have been proposed by researchers to combine autonomous code summarization with development tools.[6]

VII. CONCLUSION

Autonomous summarization of code with LLMs might bring the revolution to software development in terms of making easier the comprehension of code and its maintenance. After all, this profession has made significant progress, but there are still many obstacles to overcome and much room for further study. The survey of the literature is meant to provide a broad overview of the state of the field today while pointing out essential discoveries, difficulties, and potential paths forward. Recent works have reported that large language models (LLMs) are capable of driving a paradigm shift in code summarization toward significant advances in software engineering through the automatic generation of descriptions that are both concise and sufficiently accurate.

We have outlined in this paper a review of the state-of-the-art code summarization using LLMs, focusing on different strategies applied, key findings, and the inherent challenges.

We discuss transfer learning, an effective technique for enhancing LLM performance on the code-summarizing task. Great strides in quality were realized with the pretraining on large-scale natural language datasets, followed by fine-tuning on code-specific data.

In the same spirit, few-shot learning can thus be cited as one of the practical approaches that use larger fine-tuning datasets and smaller training datasets to prepare a model to do well with smaller numbers of supervised samples.

Automatic semantic augmentation is an idea that holds the potential to improve correctness and quality in the summaries of codes by enriching LLM prompts with semantic information that can improve context understanding. Even transformer-based models have taken boundaries further by focusing on collecting contextual information and dependencies covering very long ranges.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.423

|www.ijirset.com | A Monthly Peer Reviewed & Referred Journal | e-ISSN: 2319-8753; p-ISSN: 2347-6710|

|Volume 13, Issue 6, June 2024|

|DOI: 10.15680/IJIRSET.2024.1306164|

However, important issues remain with such a pipeline, notably: a lack of high-quality training data, code complexity, difficulty in the laborious process of binary code summarization, and handling long context scenarios. Secondly, the other major challenge is that there remains a substantial semantic gap between natural language and code, and it is yet to be bridged. Moreover, training data in the initial phases of code generation is scarce, which is quite a good challenge. Future work could look into some exciting domains; for example, using multimodal learning could enhance code summarization with combined auditory and visual modalities in natural language processing. Another critical area of research is to enhance the explainability of code summaries through mechanisms for attention and visualizations. This development can be simplified and more accessible by using plugins and APIs to integrate autonomous summarization of code into other tools used in development. The successful development of autonomous summarization of code using LLMs is bound to impact software development in a manner never seen before. This has the potential to transform how developers read and maintain code, increasing reuse, speeding up issue-find, and even easing on-boarding. However, it takes much more research and creativity to fulfill this promise about how the problems outlined could be solved.

In summary, code summarization with LLMs has advanced a lot, but there are many concerns left unanswered and much research yet to be done on this vibrant topic. The research community can pave the way for more advanced and valuable implementations of LLMs in software engineering and ultimately change the scene of code comprehension and maintenance by expanding current advancements and addressing current hurdles.

ACKNOWLEDGMENT

I would like to thank my mentor, Prof. Rashmi Gourkar for encouraging me on this research topic. Alongside, I'd like to thank MET ICS for giving me an excellent opportunity to write this research paper.

REFERENCES

1. https://www.semanticscholar.org/paper/Context%E2%80%90based-transfer-learning-for-low-resource-Guo-Chai/acd31042a2edcecc5fca8e64966240889faaed87
2. https://www.semanticscholar.org/paper/Context%E2%80%90based-transfer-learning-for-low-resource-Guo-Chai/acd31042a2edcecc5fca8e64966240889faaed87
3. https://arxiv.org/abs/2304.06815 T. Ahmed, "Automatic Semantic Augmentation of Language Model Prompts (for Code Summarization)".
4. https://arxiv.org/abs/2207.04237 T. Ahmed, P. Devanbu, "Few-shot training LLMs for project-specific code-summarization,"
5. https://arxiv.org/abs/1909.04352
6. https://arxiv.org/abs/2307.02503
7. https://arxiv.org/abs/2312.09601
8. https://arxiv.org/abs/2310.06839
9. https://arxiv.org/abs/2305.12865

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi cross ref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details