Code Summarization Assignment

Palak Rathore

January 23, 2024

Contents

Abstract **2**
0.1 Research Questions . . . 2

1 Introduction **2**

2 Datasets Viewed **2**
2.1 IRSE . . . 2
2.2 Conala-train . . . 2
2.3 CodeXGLUE . . . 2

3 Models Observed **2**

4 Performance Comparison **3**

5 Code LLMs vs Generic LLMs **3**
5.1 Code LLM . . . 3
5.2 Generic LLM . . . 3

6 Zero Shot vs Few Shot **3**

7 Performance on Different Programming Languages **3**

8 Literature Review **4**
8.1 Early Approaches . . . 4
8.2 Machine Learning-based Approaches . . . 4
8.3 Large Language Models (LLMs) . . . 4
8.4 Challenges in Code Summarization . . . 4

Conclusion **4**

Future Directions **4**

References **4**

Abstract

Automating code documentation through explanatory text can prove highly beneficial in code understanding. Large Language Models (LLMs) have made remarkable strides in Natural Language Processing, especially within software engineering tasks such as code generation and code summarization. This study specifically delves into the task of generating natural-language summaries for code snippets, using various LLMs.

0.1 Research Questions

The primary research questions guiding this study are as follows:

1. How effective are Larger Language Models (LLMs) in summarizing code snippets?
2. What is the impact of zero-shot and many-shot learning techniques on code summarization?
3. How do code-specific LLMs compare to generic LLMs in code explanation tasks?

1 Introduction

Examine Larger Language Models (LLMs) and evaluate their effectiveness in summarizing provided code snippets, specifically focusing on their capacity for code explanation. Investigate the applicability of zero-shot and many-shot learning techniques within this context. Additionally, explore the distinctions between code-specific and generic LLMs.

2 Datasets Viewed

2.1 IRSE

Information Retrieval in Software Engineering (IRSE) track at Forum for Information Retrieval Evaluation (FIRE) 2023. Each sample in the dataset is a (code snippet, code explanation) pair. The explanation is a natural language description that denotes what task the code snippet is performing. We refer to this dataset as "IRSE".

2.2 Conala-train

Publicly available conala-train dataset as a data source for few-shot and instruction fine-tuning. This dataset consists of 1666 unique samples of (code snippet, code explanation) pairs.

2.3 CodeXGLUE

CodeXGLUE code summarization benchmark. It should be noted that this dataset is unrelated to the Codex model. CodeXGLUE is originally adapted from the CodeSearchNet dataset. It's multilingual, with data from six different languages (i.e., Ruby, JavaScript, Java, Go, PHP, Python).

3 Models Observed

- Llama2-70B-Chat
- Code Llama-13B-Instruct
- Llama-2-Coder-7B
- CodeUP-13B
- StarCoder-15.5B

Model	BLEU-1	BLEU-2	BLEU-N
CodeBERT			
Llama2-70B-Chat	0.019	0.008	0.004
0.338			
CodeLlama-13B-Instruct	0.189	0.073	0.036
0.498			
Zero Shot CodeUP-13B	0.010	0.003	0.001
0.310			
StarCoder-15.5B	0.069	0.024	0.005
0.336			
Llama-2-Coder-7B	0.189	0.075	0.023
0.475			

Table 1: Performance of different LLMs

4 Performance Comparison

The table illustrates the performance of five distinct Language Model Models (LLMs) across three methodologies: zero-shot, few-shot, and zero-shot applied to the instruction fine-tuned model.

5 Code LLMs vs Generic LLMs

5.1 Code LLM

LLM like OpenAI Codex, CoPilot these are famous, and these are code LLM which means they are code-specific and they are not based on guessing the next words based on previous rather they depend on the code and answer it.

5.2 Generic LLM

A generic Language Model (LLM) refers to a language model that is trained on a diverse range of textual data without specific domain or task-oriented fine-tuning. It is designed to understand and generate human-like text across various topics and contexts. Generic LLMs are pre-trained on large datasets containing a broad spectrum of language, including general knowledge, literature, news, and more.

On experimenting on the IRSE dataset, we get insights that Code LLM's perform better than generic LLM's.

6 Zero Shot vs Few Shot

For Zero shot, Llama2 models showed good results, but once we try few shot Codex models are best; they outperform many SOTA models for a 10-shot, and the results are based on the research paper [**?**].

7 Performance on Different Programming Languages

Language	Java	Python	Ruby	JavaScript	Go	PHP	Average
CodeBERT	18.8	17.73	12.61	14.30	18.5	25.88	17.97
Codex	20.22	18.19	14.64	16.34	19.18	26.46	19.17

Table 2: Comparison of CodeBERT and Codex on different programming languages

The table shows the performance of CodeBERT and Codex on various programming languages.

8 Literature Review

Code summarization has gained significant attention in recent years due to its potential to improve code understanding and documentation. Several approaches and models have been proposed to tackle the challenges associated with generating concise and informative summaries for code snippets.

8.1 Early Approaches

Early approaches to code summarization often relied on rule-based methods and handcrafted heuristics. These methods struggled to capture the complexity and variability present in real-world codebases, limiting their effectiveness.

8.2 Machine Learning-based Approaches

The advent of machine learning, particularly deep learning, has revolutionized code summarization. Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and more recently, Transformers, have been applied to capture the sequential and structural aspects of code for summarization.

8.3 Large Language Models (LLMs)

The emergence of Large Language Models (LLMs) has marked a significant breakthrough in code summarization. Models like OpenAI Codex, GitHub Copilot, and CodeBERT have demonstrated the ability to generate human-like summaries for code snippets. These models leverage pre-training on vast amounts of code and natural language data, enabling them to understand the context and nuances of code.

8.4 Challenges in Code Summarization

Despite the advancements, code summarization faces several challenges. Code variability, ambiguity in comments, context awareness, and defining appropriate evaluation metrics are among the key challenges. Adapting models to support multiple programming languages and integrating them seamlessly into Integrated Development Environments (IDEs) are also areas of ongoing research.

Conclusion

Code summarization, facilitated by Large Language Models (LLMs), presents a promising avenue for improving code understanding and documentation. The comparative analysis of various LLMs, their performance on different programming languages, and the exploration of zero-shot and few-shot learning techniques contribute valuable insights to the field. Despite the progress, addressing challenges like code variability and ensuring effective integration with development workflows remain areas for future research.

Future Directions

The evolution of code summarization tools holds promising prospects for enhancing software development workflows. Future directions for research and development in this domain include context-aware models, adaptive code summarization, integration with IDEs, and continuous learning mechanisms.

References

1. M. Lastname et al., "Title of the Research Paper," *Journal of Code Summarization*, vol. 1, no. 1, pp. 1-20, 2023. DOI: 10.1234/jcs.2023.01.01.

2. Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). *SQuAD: 100,000+ Questions for Machine Comprehension of Text.*

3. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Agarwal, S. (2020). *Language Models are Few-Shot Learners.* arXiv preprint arXiv:2005.14165.

4. See, A., Liu, P. J., & Manning, C. D. (2017). *Get to the Point: Summarization with Pointer-Generator Networks.* Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017).