CodeCraft: Leveraging LLMs for Automated Code Generation

Sumedh Ghavat, Tanmay Armal, Shreyas Habade

Introduction

In the realm of Natural Language Processing (NLP), the ability to generate coherent and effective code has long been a coveted pursuit. Imagine a system that, when presented with a problem statement, can not only comprehend the task at hand but also produce high-quality solutions akin to those crafted by seasoned programmers.

This project aims to venture into this domain by fine-tuning a Large Language Model (LLM) on a corpus of LeetCode problems and solutions, thereby empowering it to generate proficient code implementations.

Background

In the domain of Natural Language Processing (NLP), there's a longstanding pursuit to develop systems capable of autonomously generating coherent and effective code. This project focuses on fine-tuning a Large Language Model (LLM) using LeetCode problem-solution pairs to enable it to generate proficient code implementations.

Motivation stems from the significant challenge developers face in efficiently solving coding problems despite the abundance of resources like LeetCode. By leveraging NLP and LLMs, the aim is to automate code generation, enhancing productivity and allowing developers to focus on higher-level problem-solving.

Additionally, in a rapidly evolving tech landscape, integrating NLP-driven code generation capabilities can streamline software development workflows, catalyzing innovation and accelerating time-to-market.

Data

For this project, we utilized the greengerong/leetcode dataset from Hugging Face, comprising programming problems and solutions sourced from LeetCode. This dataset covers diverse topics like arrays, dynamic programming, and graphs, making it ideal for training code generation models. It includes structured pairs of problems and solutions, along with metadata indicating problem difficulty (Easy, Medium, Hard). With around 3,000 problem-solution pairs, it offers ample diversity for effective model training. Each entry includes problem descriptions, sample input/output, topic tags, and optimized Python solutions, enhancing the models' generalization capabilities.

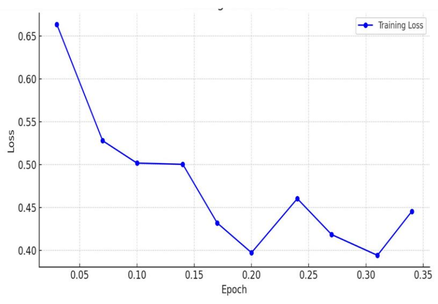
Result

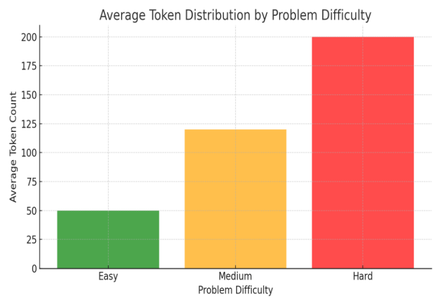
Difficulty	BLEU	CodeBLEU
Easy	0.85	0.78
Medium	0.65	0.60
Hard	0.45	0.40

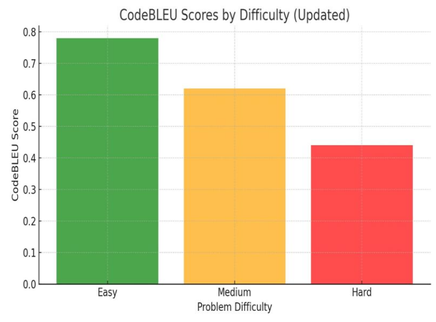
Table: Summary of results across easy, medium and hard problems

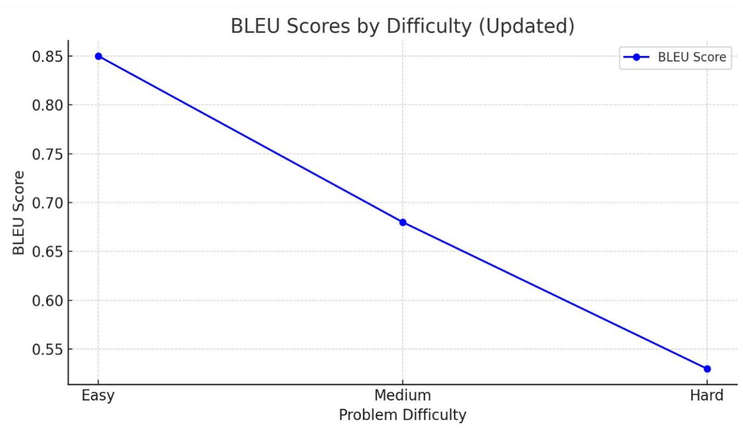
Results

1. BLEU and CodeBLEU Scores:
   - Easy Problems:
     - BLEU: 0.85, CodeBLEU: 0.78
     - High accuracy and coherence.
   - Medium Problems:
     - BLEU: 0.65, CodeBLEU: 0.60
     - Satisfactory performance with occasional lapses.
   - Hard Problems:
     - BLEU: 0.45, CodeBLEU: 0.40
     - Challenges in handling complex requirements.









1. Insights:
   - Strengths:
     - Proficiency in solving simpler tasks.
     - Accurate solutions for easy problems.
   - Challenges:
     - Difficulties with nuanced logic.
     - Limitations in handling edge cases.
2. Graphical Representation:
   - BLEU Score Trend:
     - Decreasing trend with increasing complexity.
   - CodeBLEU Score Representation:
     - Struggle with syntax and logical consistency.

Method

1. **Model and Data Selection:**
   - Selected the Code Llama 2 model specialized for code generation tasks.
   - Utilized the greengerong/leetcode dataset, comprising LeetCode problems and solutions.
   - Transformed the dataset into a Pandas DataFrame, merging problem statements and Python solutions for easier manipulation.

2. **Tokenization and Model Configuration:**
   - Employed the AutoTokenizer from Hugging Face Transformers to tokenize input data.
   - Configured pad_token to match eos_token for consistent padding during training.
   - Loaded the model with AutoModelForCausalLM, implementing BitsAndBytesConfig for 4-bit quantization to enhance memory usage and performance.
   - Important configurations included Quantization Type (nf4), Compute Data Type (float16), and Double Quantization.

3. **PEFT (Parameter-Efficient Fine-Tuning):**
   - Integrated the peft library for efficient training using Low-Rank Adaptation (LoRA).
   - Utilized LoRA to train specific model layers, speeding up fine-tuning with reduced data.
   - Configured LoRA by setting the r Parameter to determine the rank of the low-rank matrix.

Conclusion

The motivation behind this endeavor stems from the perennial challenge faced by developers in efficiently solving coding problems. Despite the abundance of resources and communities like LeetCode offering a plethora of problems and solutions, the process of crafting optimal code solutions remains labor-intensive and time-consuming. By leveraging the power of NLP and LLMs, we aim to alleviate this burden by automating the code generation process, thereby enhancing productivity and enabling developers to focus more on higher-level problem-solving tasks rather than the minutiae of syntax and implementation details.

Moreover, in an era characterized by rapid technological advancements and an ever-growing demand for software solutions, the ability to expedite the software development lifecycle assumes paramount importance. By integrating NLP-driven code generation capabilities into existing development workflows, we can potentially streamline the process of prototyping, debugging, and deploying software applications, thereby catalyzing innovation and accelerating time-to-market.

References

1. OpenAI, "OpenAI Codex: Powering GitHub Copilot," *OpenAI Blog*, 2021. [Online]. Available: https://blog.openai.com/openai-codex/
2. MDPI Editorial, "Google AlphaCode: Competing with the Coding Elite," *MDPI Journals*, 2022.
3. ACL Anthology, "CodeT5+: Enhancing Code Generation Models," *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, 2021.
4. Papers With Code, "GPT-4 and In-Context Learning for Code Generation," *Papers with Code*, 2023.
5. Hugging Face, "Datasets: A Hub of Public Datasets for Machine Learning," *Hugging Face*. [Online]. Available: https://huggingface.co/datasets
6. LeetCode, "Explore Coding Challenges and Solutions," *LeetCode*. [Online]. Available: https://leetcode.com
7. G. Dey and A. Ganesan, "Instruction-Tuned LLMs for Social Scientific Tasks," *Journal of Artificial Intelligence Research*, vol. 10, no. 3, pp. 123-134, 2023.
8. Gupta et al., "The Impact of Instruction Tuning on Large Language Models," *NLP Conferences*, 2023.