UCS748- Generative AI

CodeScribe

AI Powered Code Commenter

Jyotansh Mohindru
Parth Taggar

Submitted to: **Priya Raina**

Problem Statement & Motivation

- **The Problem:** Developers often skip writing comments due to time constraints, leading to "legacy code" that is hard to maintain and debug.
- **The Gap**: Manual documentation is slow; existing tools are often rule-based or lack context.
- **Our Solution:** A lightweight, fine-tuned AI model that automatically adds inline comments and summaries to raw code snippets.
- **Target Audience:** Beginner programmers, students, and developers handling legacy code

System Architecture (High Level)

Pipeline Overview:

1. **Dataset Curation:** Instruction-Response pairs.
2. **Fine-Tuning:** Google Colab (T4 GPU) -> CodeT5-Small Model.
3. **Export:** Saving weights (.bin/.safetensors) and Tokenizer.
4. **Deployment:** Local inference using Streamlit interface.

Dataset & Preprocessing

- **Dataset Type:** Custom Instruction-Tuning Dataset (JSONL format).
- **Size:** ~20,022 examples.
- **Structure:**
  - Instruction: "Add inline comments..."
  - Input: Raw Code (e.g., def fib(n)...)
  - Output: Commented Code + Summary.
- **Preprocessing:**
  - Unified text prompts: ### Instruction: \n {instr} \n ### Input: \n {code}.
  - Tokenization: Padding/Truncation to fixed length (256-512 tokens).

Model Selection: Why CodeT5?

- **Selected Model:** Salesforce/codet5-small.
- **Architecture:** Encoder-Decoder (Seq2Seq) Transformer.
- **Why CodeT5?**
  - Pre-trained specifically on code (GitHub), not just text.
  - Understands code semantics better than generic GPT models.
- **Why Small Variant?**
  - Lightweight (60M parameters).
  - Trainable on free Colab GPUs (T4).
  - Fast inference on local CPUs.

Training Configuration (Hyperparameters)

- **Platform:** Google Colab (NVIDIA T4 GPU, 16GB VRAM).
- **Library:** Hugging Face transformers, seq2seq Trainer.
- **Key Parameters:**
  - Batch Size: 8 (per device).
  - Learning Rate: 5e-5 (Standard for fine-tuning).
  - Optimizer: AdamW.
  - Epochs: 3.
  - Precision: FP16 (Mixed Precision) – Crucial for speed and memory efficiency.
  - Loss Function: Cross-Entropy.
-

Training Results & Observations

- **Loss Trajectory:**
  - **Initial Loss:** ~4.43 (Model guessing randomly).
  - **Final Loss:** ~1.44 (Model learned the pattern).
  - **Note:** Previous attempts had "cheating" loss (0.001) due to incorrect data formatting; this was corrected in the final build.
- **Convergence:** The model showed steady improvement over 7,509 steps.
- **Outcome:** Successfully learned to distinguish between code logic and comment syntax.

Implementation & Deployment

- **Framework:** Streamlit (Python).
- **Inference Flow:**
  - User inputs code.
  - App formats prompt (Instruction + Input).
  - Tokenizer converts to IDs.
  - Model generates output using Beam Search (Beams=4) for higher quality.
  - Decoder converts IDs back to text.
- **Offline Capability:** Runs entirely locally; no API calls or internet required after initial download.

Results & Samples

- **Input:** Recursive Factorial Function.
- **Output:** Added # Base case and # Recursive step comments correctly .
- **Input:** Loop Logic.
- **Output:** Identified loop range and operation (e.g., # Loop through numbers 0 to 2).
- **Limitations:**
  - Can struggle with very long, complex files (context limit).
  - Sometimes repeats summaries in the comments.

Future Scope

- **Multi-Language Support:** Expand beyond Python to C++, Java, and Go.
- **Advanced Docs:** Generate Docstrings (params/returns) instead of just inline comments.
- **Metrics:** Implement BLEU or ROUGE scores for quantitative evaluation.
- **IDE Extension:** Port the Streamlit logic to a VS Code Extension for real-time use.

Thank You