DocuMint: Docstring Generation for Python using Small Language Models

COSC540 Advanced Software Engineering

Bibek Poudel
Sekou Traore
Adam Cook
Shelah Ameli

Outline

- Context & Motivation
- Methodology & Experiments
- Data Extraction Operations in WoC
- Conclusion

Context & Motivation

Context & Motivation

Large Language Models for Code (Code LLMs)

⚡EvalPlus Tests⚡

#	Model	pass@1
1	🥇GPT-4-Turbo (Nov 2023)✨	⚡77.5
2	🥈claude-3-opus (Mar 2024)✨	⚡75
3	🥉CodeQwen1.5-7B-Chat✨	⚡73.8
4	DeepSeek-Coder-33B-instruct✨	⚡72.8
5	OpenCodeInterpreter-DS-33B✨💙	⚡71.2
6	Artigenz-Coder-DS-6.7B✨	⚡71.1
7	GPT-3.5-Turbo (Nov 2023)✨	⚡70.2
8	Magicoder-S-DS-6.7B✨💙	⚡70.2
9	Llama3-70B-instruct✨	⚡69.8
10	OpenCodeInterpreter-DS-6.7B✨💙	⚡69.2
11	claude-3-haiku (Mar 2024)✨	⚡68.8
12	DeepSeek-Coder-6.7B-instruct✨	⚡68.2

Base Tests

#	Model	pass@1
1	🥇claude-3-opus (Mar 2024)✨	86.2
2	🥈GPT-4-Turbo (Nov 2023)✨	85.6
3	🥉CodeQwen1.5-7B-Chat✨	81.5
4	DeepSeek-Coder-33B-instruct✨	80.8
5	OpenCodeInterpreter-DS-33B✨💙	79.8
6	GPT-3.5-Turbo (Nov 2023)✨	79.7
7	Llama3-70B-instruct✨	79.3
8	claude-3-haiku (Mar 2024)✨	78.5
9	Artigenz-Coder-DS-6.7B✨	78.2
10	Magicoder-S-DS-6.7B✨💙	78.1
11	claude-3-sonnet (Mar 2024)✨	77.2
12	OpenCodeInterpreter-DS-6.7B✨💙	77

Code LLMs

https://evalplus.github.io/leaderboard.html

Context & Motivation

LLMs ❤️ Code

code llama

New Open model

CodeGemma

Code generation and assistance

</> StarCoder

May The Source Be With You!

Context & Motivation

LLMs ❤️ Code

- Formal Syntax
- Deterministic output
- Limited Vocabulary

code llama

New Open model

CodeGemma

Code generation and assistance

</> StarCoder

May The Source Be With You!

Context & Motivation

Open source

OpenDevin: Code Less, Make More

🚀 Devika - Agentic AI Software Engineer 👩‍💻

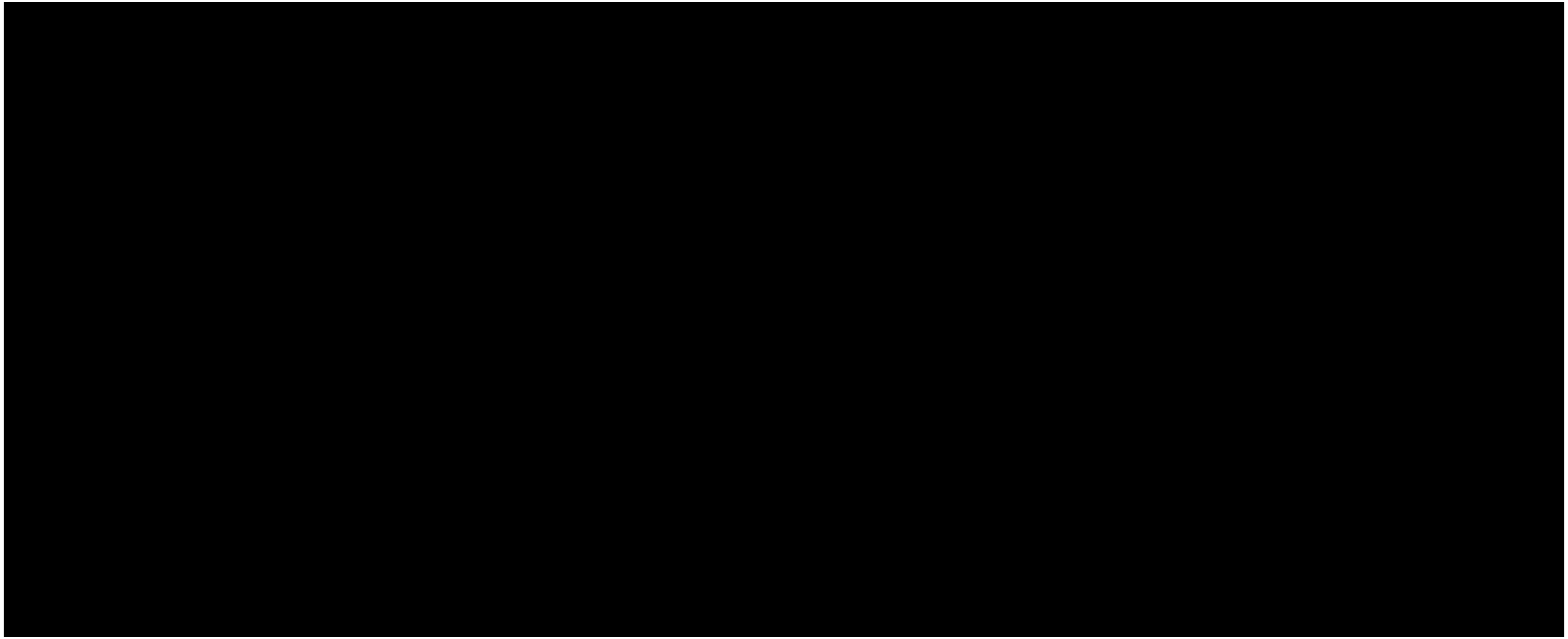
Closed source

GitHub Copilot

Ghostwriter

Devin

Context & Motivation



Context & Motivation

46%

of all code written (all programming languages) was assisted by GitHub Copilot

https://github.blog/2023-02-14-github-copilot-for-business-is-now-available/

Context & Motivation

Agents

OpenDevin: Code Less, Make More

Devin

GitHub Copilot

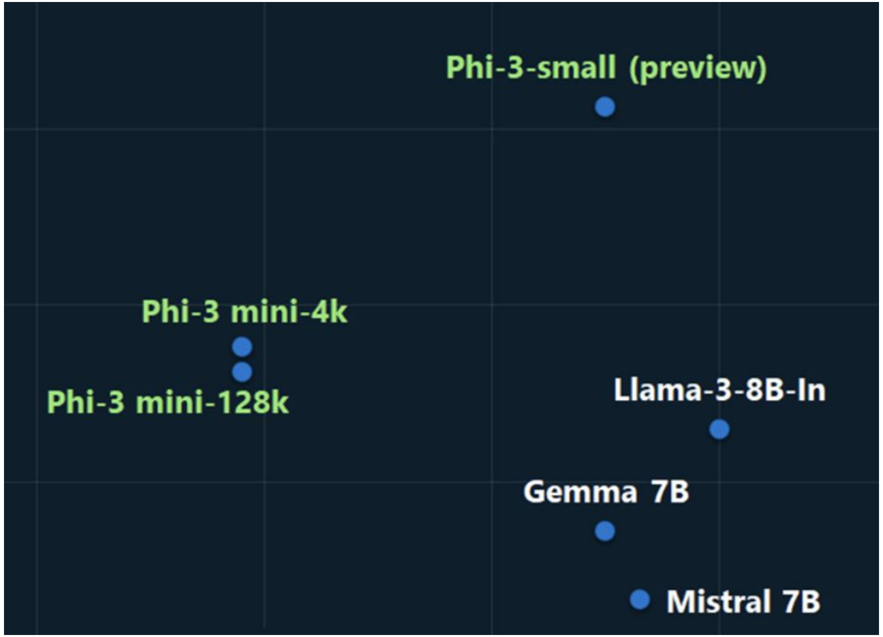
🚀 Devika - Agentic AI Software Engineer 👩‍💻

Humans

Context & Motivation

LLMs ⟹ Small Language Models (SLMs)

- Generally < 7B parameters
- Consumer CPU: ~2B params
- Consumer GPU: ~7B params



Context & Motivation

LLMs ⟹ Small Language Models (SLMs)

- Generally < 7B parameters
- Consumer CPU: ~2B params
- Consumer GPU: ~7B params

SLMs offer

- Easier access, privacy
- Low inference cost, time
- Perfect for agents!

Methodology
&
Experiments

Objective

Objective 1: Benchmark mathematically

Objective 2: Benchmark qualitatively

Objective 3: Fine-tune SLM for docstring generation

```python
def example_function(param1, param2):
    """
    This is an example of a docstring
    for a Python function.

    Docstrings are enclosed in triple
    quotes and appear immediately
    after the function definition.
    """

    # Function implementation
```

Evaluation Metrics

- **Accuracy**
  - Measures coverage of code elements
  - BERT Encoding + Cosine Similarity (BERT Score)
- **Conciseness**
  - Measures conveyance of information
  - Compression Ratio
- **Clarity**
  - Measures readability of the docstring
  - Flesch-Kincaid Readability Score

Experiment 1: MATH Benchmarks

"Use 4 SLMs to generate docstrings for input functions and **calculate the evaluation** metrics."

Experiment 1: Math Benchmark

- “Metric Quantification”
- Compared docstrings generated by SLMs vs. Claude3
  - Experiment vs. “Expert”
- Python scripts calculate numerical values
  - Accuracy: [0, 1] (⬆)
  - Conciseness: [0, 1] (⬇)
  - Clarity: [0, 100] (50 - 70)

Small Language Models (SLMs)

- Four code-based SLMs loaded using HuggingFace (Transformers)
  - **CodeGemma 7B Instruct**
  - **Meta Llama 3 8B Instruct**
  - **DeepSeek Coder 6.7B Instruct**
  - **StarCoder2 7B**
- Three 2B SLMs planned
  - CodeGemma 2B Instruct
  - DeepSeek Coder 1.3B Instruct
  - StarCoder2 3B
- Control model
  - **Claude3 Opus**

Function Data

- Selected 7 functions each from 3 widely used datasets for inference
- MBPP (Mostly Basic Programming Problems)
    - Collection of 974 simple, easy-to-solve coding problems
    - Covers a wide range of basic programming concepts and data structures
- HumanEval:
    - Contains 164 medium-level coding problems
    - Problems require a deeper understanding of algorithms and problem-solving
- Apps:
    - Consists of 10,000 of the most challenging and complex coding problems
    - Sourced from real-world software development scenarios

Experiment 1: Math Benchmarks

	Model	Accuracy	Conciseness	Clarity
7B Models	CodeGemma Instruct	0.609	0.569	76.49
	DeepSeek-Coder Instruct	0.679	0.734	64.44
	Llama 3 Instruct	0.668	0.605	64.88
	StarCoder2	0.626	0.510	69.74

- **Llama 3** performs adequately for **Accuracy, Conciseness,** and **Clarity**
  - High BERTScore, balanced ratio, balanced reading score
- **DeepSeek** performs the best for **Accuracy**, adequately for **Clarity**
  - Has the highest BERTScore
- **CodeGemma** performs the best for **Conciseness**
  - Compression ratio cannot be too low

Experiment 2: Human Benchmarks

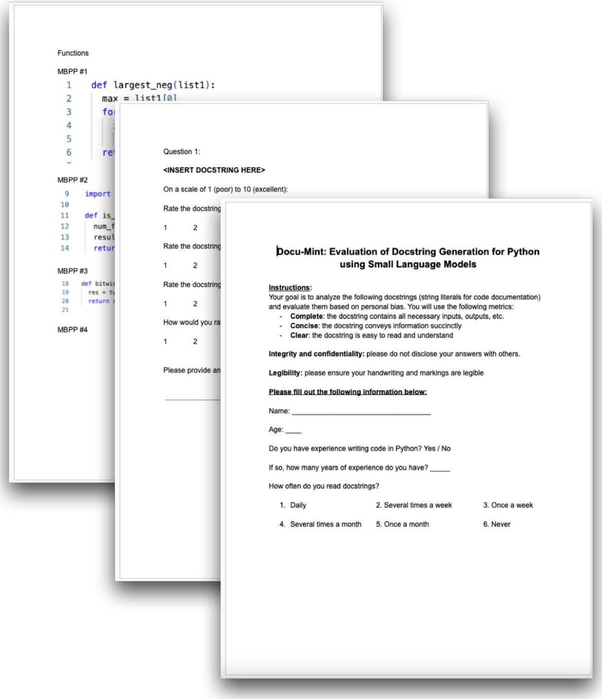
"Use 4 SLMs to generate docstrings for input functions and ask **humans to evaluate** them."

Experiment 2: Human Benchmarks

- “Metric Qualification”
- Assess the quality of the docstring generated by SLMs
- Criteria:
  - **Accuracy**: Does the docstring contain all necessary inputs and outputs?
  - **Conciseness**: Does the docstring convey information succinctly?
  - **Clarity**: Is the docstring easy to read and understand?

Experiment 2: Human Benchmarks

- Paper questionnaire given out to select individuals
  - 3 docstrings per SLM
  - Rate evaluation metrics on Likert scale
    - 1 = Docstring does not meet this metric at all
    - 10 = Docstring meets this metric very well
- *In Progress



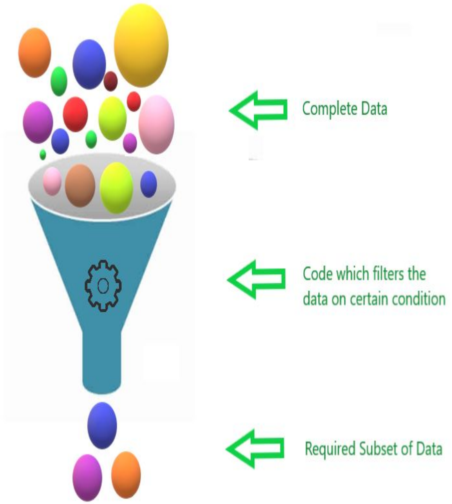
Experiment 3: Fine-tuning with WoC data

"Use World of Code data to Fine-tune a SLM for Docstring generation."

Experiment 3: Fine-tuning with WoC data

Data Sampling

- Goal: **extract 100k python** functions and docstring pairs for fine tuning
  - Exposure of the model to a diverse and qualitative set of elements
  - Filtering using baseline metrics to get as close as possible to our goal
- Filtering:
  - Metrics
  - Model
- Extraction
  - Extraction
  - Generation



Experiment 3: Fine-tuning with WoC data

Data Filtering

- Used WoC to set baseline metrics
    - Filtered projects by:
        - Number of stars > **35k**
        - Number of commits > **5k**
        - Number of forks > **10k**
        - Number of contributors > **50**
    - Parsed the projects metadata for their projectID’s(project name)

Experiment 3: Data Extraction and Cleaning

- Developed a parser to extract the pairs and generate the dataset
  - Use of the ast, gitpython and JSON modules
  - Clones repos based on their address into a purposed directory
  - Traverses the ast to extract function codes and docstrings
- Creating the dataset
  - Uses the JSON module to directly write the pairs to the dataset
  - Dataset is dumped into an output JSON file in the desired format

Experiment 3: Data Considerations and insights

- Insights in the FLOSS ecosystem
  - The most popular repos are not necessarily the most well written
  - Preliminary phase: **148k functions** reduced to **66k**
  - Ended up with **~13 GBs** of files that had syntactic errors and could not parsed with ast
- Considerations
  - nearly impossible to avoid repetitions
  - Wide discrepancies between number of functions from different projects
  - Many mega projects and frameworks are written in multiple languages even though the dataset contains large python majority files (Django, Flask, Numpy, TensorFlow etc...)

Experiment 3: Fine Tuning

Fine Tuning

- Model: CodeGemma 2B Instruct
- Method: LoRA (around 34,000,000 parameters)
- Data: 100,000 functions + docstrings from WoC
- Evaluation: Against the original model from Google
- Training time: 5 days (RTX 3090)

Conclusion

Conclusion

- Motivations
- Experiment 1
- Experiment 2
- Experiment 3
- Report Writing (50% complete)

Thank You!