Examples out of Thin Air
AI-generated Dynamic Context to Assist Program Comprehension by Example

Toni Mattis
Eva Krebs
Martin Rinard
Robert Hirschfeld

Software Architecture Group
HPI, University of Potsdam, Germany

CSAIL
MIT, Cambridge, MA, USA

PX/24 | Lund, Sweden | March 11
Seeing Concrete State & Behavior

Code

Abstract

{...

...

}

Data

Concrete

Behavior

Concrete

Debugger

REPL / Notebooks

Example-based Live Programming

Mattis, Krebs, Rinard, Hirschfeld | AI-generated Examples | HPI Software Architecture Group & MIT CSAIL
Babylonian Programming | Babylonian/S

Example

Probes

Classes

Methods

Example

Method Source
Examples are created manually!

**Goal:**
Automatically get [part of] a running program into a state that illustrates its behavior

**Opportunity:**
Use “AI” (large language models) to generate such a state
What’s an LLM?

```java
WebClient new httpGet: 'https://www.example.com'
```

Language Model

- 0.5 https
- 0.4 http
- 0.1 www
What’s an LLM?

WebClient new httpGet: ‘

Language Model

[0.9 :// ]
[ 0.1 . ]
[ 0.0 - ]
What’s an LLM?

Large Language Model*

Billions of parameters
Terabytes of training data

*) In this work: CodeLlama-7B
Approach

LLMs can generate **code** but not **run-time state** (yet)
Approach | Limitations

- Maximum context size (~250 LOC)
- Language Model
- Training data
- No project-specific knowledge
- Mixed versions and distributions
- Out of date
Approach | **Fine-tuning**

**Fine-tuning** (⏰ 1h)

- **Codebase**
  - Code

  **Adapter**
  - Prompt
  - LLM

  **Test Case**
  - Execution

**Generation** (⏰ 3 sec.)

- Standard library
- Coding styles/practices
- Project-specific abstractions
Approach | **Fine-tuning**

**Class 1**
- Method 1
- Method 2

**Class 2**
- Method 3

**Problem structure:**
- LLM completes left-to-right
- Code
- Example

**Fine-tuning Data**

- Class 1
- Method 1
- Method 2
- Class 2
- Method 3

- calls
- uses
Next Steps | Babylonian Integration

- Study usefulness
- Interactive refinement
Next Steps | Directly to Objects

**Goal:** Remove indirection over test code

Training data & prompt

- Serialization
- Context (re-)construction
Repository and Community Resources

- Examples from documentation, mailing lists, ...
- Ethical/legal considerations when using community content for training (plagiarism, attribution, consent, ...)
Summary

- (Fine-tuned) LLM that generates examples in Squeak/Smalltalk
- LLMs generate abstract code, live programming is about concrete data and behavior and feedback

How can we better co-design LP + LLMs?
Backup Slides
Architecture Overview

Diagram:
- **Context**: WebClientExample >> performHttpGet
- **Promp**: Language Model
  - **Adapter**: Pre-trained LLM
  - **Completion**: WebClient new httpGet: ‘https://...’
- **Serialization**
- **Compile & Run**: Method of Interest
- **Code and Test Cases**

**Training Data**
- Base System
  - Program
  - Call Graphs
- Program Analysis

**Program Analysis**

**Method of Interest**
Fine-tuning

- Low-rank Adaptation (LoRA)
- Train <1% of model weights

- Filtering (80% of Methods)
- Insert redundant definitions
Prompt Construction

- Follow Dependencies
  - Call Graph
    - Dynamically sampled for project
    - Statically approximated for rest of environment
  - Data dependencies
    - Symbolic execution → Data flow graph
    - Link inter-method DFGs to a full DFG with help of call graph
- Include immediate dependents
- **Future work**: Ablation study
Greedy vs. Search

Language Model

WebClient new httpGet:

https ://

0.5 https
0.4 http
0.1 www

0.9 ://
0.1 .
0.0 -

0.3 www
0.2 squeak
0.1 \n
0.9 <EOS>

0.1 ...

P = 0.05
P = 0.4
P = 0.05

P = 0.45
P = 0.3
P = 0.09

P = 0.14
P = 0.3
P = 0.06

P = 0.04
P = 0.08

P = 0.01
Example Prompt

```smaller
<class>
Object subclass: #WebClient
  instanceVariableNames: 'flags server scheme timeout userAgent ...'
  classVariableNames: 'DebugLog FlagAcceptCookies FlagAllowRedirect ...'
  category: 'WebClient-Core'
</class>

<method>
WebClient >> httpGet: urlString do: aBlock
  "GET the response from the given url"
</method>

<method>
WebClientTest >> testHttpGetDo
  "Create a WebClient instance and call httpGet:do: with an example"
| client |
client := WebClient new.
client httpGet: 'http://www.squeak.org'
do: [:request |
  self assert: request method = 'GET'].
</method>
```
Benchmark

CodeLlama-7B-hf @ RTX4090

parallel generations: 1

<table>
<thead>
<tr>
<th>Tokens generated</th>
<th>64</th>
<th>128</th>
<th>192</th>
<th>256</th>
<th>320</th>
<th>384</th>
<th>448</th>
<th>512</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>128</td>
<td>128</td>
<td>192</td>
<td>256</td>
<td>320</td>
<td>384</td>
<td>448</td>
<td>512</td>
</tr>
<tr>
<td>out</td>
<td>1.2</td>
<td>2.3</td>
<td>3.5</td>
<td>4.7</td>
<td>5.9</td>
<td>7.1</td>
<td>8.3</td>
<td>9.5</td>
</tr>
<tr>
<td>128</td>
<td>1.2</td>
<td>2.4</td>
<td>3.6</td>
<td>4.8</td>
<td>6.0</td>
<td>7.2</td>
<td>8.4</td>
<td>9.6</td>
</tr>
<tr>
<td>256</td>
<td>1.2</td>
<td>2.4</td>
<td>3.6</td>
<td>4.9</td>
<td>6.1</td>
<td>7.3</td>
<td>8.5</td>
<td>9.8</td>
</tr>
<tr>
<td>384</td>
<td>1.2</td>
<td>2.4</td>
<td>3.6</td>
<td>4.9</td>
<td>6.1</td>
<td>7.4</td>
<td>8.6</td>
<td>9.9</td>
</tr>
<tr>
<td>512</td>
<td>1.3</td>
<td>2.5</td>
<td>3.7</td>
<td>4.9</td>
<td>6.1</td>
<td>7.4</td>
<td>8.6</td>
<td>9.9</td>
</tr>
<tr>
<td>640</td>
<td>1.3</td>
<td>2.5</td>
<td>3.7</td>
<td>5.0</td>
<td>6.2</td>
<td>7.5</td>
<td>8.7</td>
<td>10.0</td>
</tr>
<tr>
<td>768</td>
<td>1.3</td>
<td>2.6</td>
<td>3.8</td>
<td>5.1</td>
<td>6.3</td>
<td>7.5</td>
<td>8.8</td>
<td>10.1</td>
</tr>
<tr>
<td>896</td>
<td>1.3</td>
<td>2.6</td>
<td>3.8</td>
<td>5.1</td>
<td>6.3</td>
<td>7.6</td>
<td>8.9</td>
<td>10.1</td>
</tr>
<tr>
<td>1024</td>
<td>1.4</td>
<td>2.6</td>
<td>3.9</td>
<td>5.1</td>
<td>6.4</td>
<td>7.7</td>
<td>9.0</td>
<td>10.3</td>
</tr>
</tbody>
</table>

parallel generations: 4

<table>
<thead>
<tr>
<th>Tokens generated</th>
<th>64</th>
<th>128</th>
<th>192</th>
<th>256</th>
<th>320</th>
<th>384</th>
<th>448</th>
<th>512</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>128</td>
<td>128</td>
<td>192</td>
<td>256</td>
<td>320</td>
<td>384</td>
<td>448</td>
<td>512</td>
</tr>
<tr>
<td>out</td>
<td>1.3</td>
<td>2.6</td>
<td>3.9</td>
<td>5.2</td>
<td>6.6</td>
<td>8.0</td>
<td>9.3</td>
<td>10.7</td>
</tr>
<tr>
<td>128</td>
<td>1.4</td>
<td>2.7</td>
<td>4.1</td>
<td>5.5</td>
<td>6.9</td>
<td>8.3</td>
<td>9.8</td>
<td>11.4</td>
</tr>
<tr>
<td>256</td>
<td>1.5</td>
<td>2.9</td>
<td>4.3</td>
<td>5.7</td>
<td>7.2</td>
<td>8.8</td>
<td>10.3</td>
<td>12.0</td>
</tr>
<tr>
<td>384</td>
<td>1.6</td>
<td>3.1</td>
<td>4.5</td>
<td>6.0</td>
<td>7.6</td>
<td>9.2</td>
<td>10.9</td>
<td>12.4</td>
</tr>
<tr>
<td>512</td>
<td>1.7</td>
<td>3.2</td>
<td>4.8</td>
<td>6.4</td>
<td>8.0</td>
<td>9.7</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>640</td>
<td>1.9</td>
<td>3.4</td>
<td>5.0</td>
<td>6.7</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>768</td>
<td>2.1</td>
<td>3.7</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>896</td>
<td>2.1</td>
<td>3.7</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>1024</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
</tbody>
</table>

parallel generations: 16

<table>
<thead>
<tr>
<th>Tokens generated</th>
<th>64</th>
<th>128</th>
<th>192</th>
<th>256</th>
<th>320</th>
<th>384</th>
<th>448</th>
<th>512</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>128</td>
<td>128</td>
<td>192</td>
<td>256</td>
<td>320</td>
<td>384</td>
<td>448</td>
<td>512</td>
</tr>
<tr>
<td>out</td>
<td>1.7</td>
<td>3.5</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>128</td>
<td>1.7</td>
<td>3.5</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
</tbody>
</table>

2024-03-11

Mattis, Krebs, Rinard, Hirschfeld | AI-generated Examples | HPI Software Architecture Group & MIT CSAIL
Training Parameters

- 292k LOC $\Rightarrow$ 5.5M tokens $\Rightarrow$ 5.3k $\times$ (4 $\times$ 256-token)-batches
- 2 epochs á 20 min (~ 40 min. training)

```
100%|██████████| 5318/5318 [19:54<00:00, 4.45it/s]
epoch=1: train_epoch_loss=tensor(1.0421) eval_epoch_loss=tensor(1.0421)
```

```
lr = 3e-4
optimizer = torch.optim.AdamW(model.parameters(), lr=lr)

lr_scheduler = get_linear_schedule_with_warmup(
    optimizer=optimizer,
    num_warmup_steps=0,
    num_training_steps=(len(train_dataloader) * num_epochs),
)
```

```
[15]: peft_config = LoraConfig(
    task_type=TaskType.CAUSAL_LM,
    inference_mode=False,
    target_modules=['q_proj', 'v_proj'],
    r=16,
    lora_alpha=32,
    lora_dropout=0.05,
    bias="all"
)
```
Data Cleaning (Number of Methods)

- **passed:** 66020 (81.13%)
- **denied:** 546 (0.67%)
- **empty:** 223 (0.27%)
- **exceeded blocks:** 1616 (1.99%)
- **exceeded class variables:** 6 (0.01%)
- **exceeded classes:** 416 (0.51%)
- **exceeded keywords:** 1197 (1.47%)
- **exceeded lines:** 4561 (5.6%)
- **exceeded variables:** 1075 (1.32%)
- **too many assertions:** 1129 (1.39%)
- **too few letters:** 4586 (5.64%)
- **total LOC:** 292330
- **inserted 4806 repeated class definitions**

Manual Blacklist (Fonts)
Nesting/Complexity
Low Coupling
Arguments <= 3
Low State
Small Tests
Magic numbers/Base64/...
Redundancy
Hardware

- NVidia RTX 4090, 24GB Video Memory
  - CodeLlama uses 14GB for inference, 23GB for training
- Ryzen 7 7800X3D
- 64GB DDR5 Memory
- PCIe 4.0 SSD