Programming Puzzles

Tal Schuster, Ashwin Kalyan, Oleksandr Polozov, Adam Tauman Kalai

The P3 dataset

- A large (and growing) collection of Python puzzles.
- Currently has 397 problems:
  - **Comprehensive in domain**: Algebra; Basic Python programming; Chess; Classic puzzles; Programming challenges; Compression; Conway’s game of life; Games; Graphs; ICPC; IMO; Number Theory; Probability; ...
  - **Comprehensive in required algorithmic tools**: Recursion; Linear programming; Dynamic programming; Convex optimization; Sorting; Graph search; Programming language specific operations (such as string manipulations); ...
  - **Comprehensive in difficulty**: From trivial puzzles to major open, prize offered, algorithmic and math problems.

Enumerative solvers

- Iterate over ASTs by their learned likelihood.
- **Uniform**: unparameterized; iterating from shortest to longest.
- **Random forest**: puzzles are encoded as bag-of-rules.
- **Transformer**: pretrained RoBERTa to encode puzzles and rules.

Language model solvers

- Using GPT-3 or Codex, generate solutions as strings.
  - **Prompts**:
    - **Short**: zero-shot.
    - **Medium**: five-shot (using 5 tutorial puzzles).
    - **Long**: five-shot + English descriptions to the puzzles.
  - **Bootstrap**: starts from zero-shot and adds examples to the prompt as they are found.

User study

- 21 participants (10 beginners; 11 experienced) were given up to 6 minutes per puzzle to solve 30 puzzles.
- Very positive feedback.
- Positive correlation in perceived difficulty between humans and AI.

Example of Python puzzles

- # Find a string that when reversed and concatenated with “world” gives “Hello world”:
  ```python
def f(y: str): return y[::-1] + "world" == "Hello world"
```
- # Tower of Hanoi, often teaches recursion. Move [i, j] means move top disk on tower i to j, with 1 ≤ i, j ≤ 3:
  ```python
def f2(moves: List[List[int]], num_disks: int) -> List[List[int]]:
    state = [1] * num_disks # All disks start at tower 1.
    for i, j in moves:
        assert state.index(i) <= (state + 1, 2, 3).index(j), "bigger disk on top"
        move = [state.index(i)] # Move smallest disk from tower i to tower j.
        state = [3] + num_disks # All disks must end on tower 3.
```
- # Find a non-trivial integer factor d of a large number n:
  ```python
def f3(n: int, m: int) -> None:
    if n <= m:
        return
    if n % d == 0:
```

Valid solution to £2 generated by Codex-cushman (Med. prompt):

```python
def g(num_disks=6):
    def hanoi(n, p, q):
        if n > 0:
            hanoi(n-1, p, r)
            move = [r, p, q]
            hanoi(n-1, q, p)
    for i in range(num_disks):
        assert f2(moves, num_disks)
        return moves
```

Results

- **Significant room for improvement for autocompletion (small k)**
- **Codex-davinci results**
  - Very positive feedback.
  - Positive correlation in perceived difficulty between humans and AI.
  - Puzzles evaluate the algorithmic proficiency of AI models and allow comparisons against human coders.