Learning to Infer Program Sketches

MohammadAmin Raeisi

Summary

This paper proposes a novel neuro-symbolic synthesizer that generates codes from examples or natural language instructions given by users. When the programming task is hard or the programmer is a beginner, the programmer would have to reason about the entire program structure from the first principles and would not recognize any programs. But When the programming task is easy, the programmer can recognize most of the general structure of the program from learned experience and would have reason about fewer details. This paper uses this idea and designs a model which learns to generate flexible program sketches based on the difficulty of the programming task. When given an easy programming task, it should rely on its learned pattern recognition, and output a fuller program with a neural network, so that less time is required for synthesis. In contrast, when given a hard task, the system should learn to output a less complete sketch and spend more time filling in the sketch with search techniques.

Its system has two main components: a sketch generator and a program synthesizer. The sketch generator is a distribution over program sketches given the program specification. The generator is parametrized by a recurrent neural network, and is trained to assign high probability to sketches which are likely to quickly yield programs satisfying the program specification when given to the synthesizer. It uses a sequence-to-sequence RNN with attention model that encodes program specifcation via LSTM encoders and then decodes a program token-by-token while attending to the program specification. The program synthesizer takes a sketch as a starting point, and then performs an explicit symbolic search to fill the holes in order to find a program which satisfies the program specification. It has two components: a breadth-first probabilistic enumerator, which enumerates candidate programs from most to least likely, and a neural recognizer, which uses the specification to guide this probabilistic enumeration.

Strengths

- The introduction section, explains the motivation well by providing an example and has stated the weakness of the previous methods.
- It uses neural network techniques to generate more flexible sketches instead of using static, hand-designed intermediate sketch grammars, which do not allow the system to learn how much to rely on pattern recognition and how much to rely on symbolic search.
- The figure at the top of page 4 shows the overall system structure well.
- The diagrams at the end of the article clearly show that the results of this work is better than the previous works.
- It is good that the results were tested on three different datasets.

Weaknesses

- It doesn't describe the various components of the system in detail and explains more in general terms.
- It doesn't use enough examples to explain the function of different system components.
- It's better to explain the objective functions presented in section 3.2 more.
- The explanation of section 4.1 is dumb for someone who is not familiar with LSTM and attention mechanism, and it's better to explain more.
- The text of some sections is not very expressive.
- The pseudo-code of algorithm number one seems very incomprehensible at first glance, and it would be better if there were more explanations for it.
- It's better to introduce the DSL of the tested datasets.
- It's better to include the GitHub link of the tool they have implemented in the paper.