RobustFill: Neural Program Learning under Noisy I/O

MohammadAmin Raeisi

Summary

This paper presented a novel variant of an attentional RNN architecture for program synthesis, which achieves 92% accuracy on a real-world PBE task for string transformations similar to FlashFill, significantly better than previous classical rule-based and neural synthesis systems. It compares two approaches: neural program synthesis and neural program induction. Both methods are most robust from classical rule-based program synthesis, where formal grammar derives a program from a well-defined specification. Also, it shows that neural models can be trained to remain very robust to the type of noise expected in real-world data (e.g., typos) while a highly-engineered rule-based system fails.

The program synthesis approach trains a neural model that takes inputs and generates program P as token-by-token output. It needs to define a DSL of string transformation first. It is trained and fully supervised on a large corpus of pairs of synthetic I/O examples and their respective programs. The program induction approach trains a neural model which takes input-output pairs and new string x as input and generates corresponding output y, character-by-character. The induction model does not explicitly use program P at training or test time. Instead, it induces a latent representation of the program.

Next, the paper introduces some architectures for the program synthesis approach neural network and compares the accuracy between them, and the induction approach contrast rule-based methods with different metrics and values for Beam and concludes that synthesis systems have an advantage when evaluating if all outputs are correct, while induction systems have strength when evaluating which system has the most correct outputs.

Strengths

- The accuracy has reached 92% from 34% of the previous best neural synthesis approach, which is a significant improvement.
- RobustFill works on noisy data, while highly-engineered rule-based tools don't.
- Compared to previous classic methods, it is a newer method that uses neural network concepts for program synthesis.
- In the overview section, the difference between the program induction and program synthesis approaches is well explained.
- The difference between the approaches that have been investigated and their accuracy has been well shown in the diagrams.

Weaknesses

- It needs to pay more attention to the details of different methods. Most of its approach is to compare various methods with each other rather than to open and explain in detail the algorithm and mechanism of each of them. It could have increased the paper's accuracy by increasing the number of pages!
- Many repetitive concepts have been mentioned in the abstract and introduction sections.
- Figure 1 is on the first page, but there is no mention of it on that page, and it would have been better to move it to the next page.
- It has tried a lot to explain the weakness of previous classical rule-based methods and compare them with RobustFill, but it hasn't tried much to explain the weakness of the last neural systems and the reason for the need for RobustFill.
- In the overview section, despite the good explanations, it was better to use examples to convey the concept better.
- There is a typo mistake at the end of page 3. i_i is written instead of i_1.
- It needs to explain the DSL of its neural program synthesis approach more precisely and could be more specific.

- Section 4 must be more specific for people unfamiliar with the attention mechanisms.
- The network architecture of the program induction approach hasn't been explained to compare with other systems.
- It was unnecessary to mention the program induction method because it neither gives a detailed description nor performs better than the program synthesis method (in most metrics).
- It didn't try to use examples to convey concepts.