Towards Human-Like Program Synthesis

Rishabh Singh
Machine Learning as Program Synthesis

Specification (Data) → Program

More Complex Tasks
Strong Generalization
Interpretability

Democratize Programming
Search-based Program Synthesis

Tremendous Progress

- Constraint-based Synthesis (Sketch)
- Version-space Algebra (FlashFill)
- Enumerative Search (Transit)

Challenge: Scalability, Generalizability

Rajeev Alur, Rishabh Singh, Dana Fisman, Armando Solar-Lezama. CACM 2018
Human Programmers

Spec
- I/O Examples
- Natural Language
- Partial Programs

Logic Basics Experience Samples

```python
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)

def fibonacci(n):
    if n == 0:
        return 1
    if n == 1:
        return 1
    return fibonacci(n-1) + fibonacci(n-2)
```
Program Synthesis vs Differentiable Programming

Neural Programmer-Interpreters
Scott Reed & Nando de Freitas
Google DeepMind
London, UK
scott.ellison.reed@gmail.com
nandodefreitas@google.com

Neural Turing Machines
Alex Graves
gregwayne@google.com
Ivo Danihelka
danihelka@google.com
Google DeepMind, London, UK

Inferring Algorithmic Patterns with Stack-Augmented Recurrent Nets
Armand Joulin
Facebook AI Research
770 Broadway, New York, USA.
a.joulin@fb.com

Tomas Mikolov
Facebook AI Research
770 Broadway, New York, USA.
t.mikolov@fb.com

Neural Random-Access Machines
Karol Kurach* & Marcin Andrychowicz* & Ilya Sutskever
Google
{kkurach, marcina, ilyasu}@google.com
Neural Random Access Machines

Differentiable Semantics

Karol Kurach, Marcin Andrychowicz, Ilya Sutskever et al. ICLR 2016
Program Induction vs Synthesis

**Program Induction**
- Differentiable computation
- Generalization Challenges
- Lots of Examples
- Single-task Learning
- Non-interpretable programs

**Program Synthesis**
- Functional Abstraction
- Better Generalization
- Fewer Examples
- Multi-task Learning
- Interpretable Programs
Human-like Programming vs Search

1. Intuition vs Enumeration
2. Improvements with Experience
3. Multi-modal Specifications
4. Sub-problems + Compositions
5. Mistakes vs Completely correct
Human-like Programming

1. Intuition vs Enumeration
2. Improvements with Experience
3. Multi-modal Specifications
4. Sub-problems + Compositions
5. Mistakes vs Completely correct
Human-like Programming

1. Intuition vs Enumeration
   No Super-human computation!

2. Improvements with Experience

3. Multi-modal Specifications

4. Sub-problems + Compositions

5. Mistakes vs Completely correct
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interested teammates</td>
<td>Office address</td>
<td>E-mail</td>
<td>Message greeting</td>
<td>Zip code</td>
<td>Paid/Not Paid</td>
</tr>
<tr>
<td>Regina King</td>
<td>500 W 2nd St, Austin, TX 78701</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yahya Abdul-Mateen II</td>
<td>500 W 2nd St, Austin, TX 78701</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Andrew Howard</td>
<td>2930 Pearl Street, Boulder, CO. 80301</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tom Misson</td>
<td>1160 Bordeaux Drive, Sunnyvale, CA 94089</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sara Vickers</td>
<td>1160 Bordeaux Drive, Sunnyvale, CA 94089</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jeremy Irons</td>
<td>2930 Pearl Street, Boulder, CO. 80301</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jean Smart</td>
<td>1160 Bordeaux Drive, Sunnyvale, CA 94089</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tim Blake Nelson</td>
<td>1160 Bordeaux Drive, Sunnyvale, CA 94089</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Louis Gossett Jr.</td>
<td>111 8th Ave. New York, NY 10011</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>James Walk</td>
<td>500 W 2nd St, Austin, TX 78701</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Danny Boyd Jr.</td>
<td>2930 Pearl Street, Boulder, CO. 80301</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Don Johnson</td>
<td>111 8th Ave. New York, NY 10011</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dylan Schombing</td>
<td>1160 Bordeaux Drive, Sunnyvale, CA 94089</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frances Fisher</td>
<td>2930 Pearl Street, Boulder, CO. 80301</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lilly Rose Smith</td>
<td>1160 Bordeaux Drive, Sunnyvale, CA 94089</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adelynn Spoon</td>
<td>2930 Pearl Street, Boulder, CO. 80301</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Steven G. Norfleet</td>
<td>2930 Pearl Street, Boulder, CO. 80301</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hong Chau</td>
<td>500 W 2nd St, Austin, TX 78701</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jacob Ming-Trent</td>
<td>111 8th Ave. New York, NY 10011</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Neuro-symbolic Program Synthesis

DSL → Training Data Generator → Synthesizer

Neural Model + Program Semantics

E. Parisotto, A. Mohamed, R. Singh, L. Li, D. Zhou, P. Kohli. ICLR 2017
Synthetic Dataset Generation

Reference program: GetToken_Alphanumeric_3 | GetFrom_Colon_1 | GetFirst_Char_4

| Ud 9:25,JV3 0bb | 2525,JV3 bbUd92 |
| zLny zmHg 8:43 A44q | 843 A44qzLny |
| A6 g45P 10:63 Jf | 1063 JfA6g4 |
| cuL.zF.dDX,12:31 | dDX31cuLz |
| ZiG OE bj3u 7:11 | bj3u11ZiG0 |

Reference program: GetToken_WS_1(GetSpan(Number, 1 End, ‘/’, 3, End)) | Const(‘R’) | GetToken_Word_5 | Const(‘L’) | Const(‘,’) | ToProper(GetToken_Word_3) | GetToken_Alphanumeric_5 | EOS

| aC Ic 3.rFL jiW.MmB fzYoa TX oNpV fHm /ai WHGM | JUF RMmBL,RVKW |
| Pgso.0Xp VKW Jo R9 0JUF / / Xir | |
| wa.Xvq-wo-isxn KD.qxpkH mACHu/ZNI | fyARKDL,TdnAB |
| Qhs-DAr,UAr-UcP.Ps xjK-JL0,AB.tdn,1-fyA//eZ | |
| Iceg gbeOz ck CbwoZ /Zmfb WMyoO /10 CQIXs,EkeFJAxı | Eŷ xAG RZmfB,L,qvqqEŷ |
| Ld a9z aSd Cse9 Ey xAG /QVqq njc ukx | |
| qm/CsPc oaSUw,wKz.rRH,jFq0.PGihT IE-2,NL | XeLrrRHL,QnXel |
| zzToV-2W6z,dE,Ptl /dSZR.XeI/eya-EA-qN kf.Yo | |
| wUx -7.ND7.xIE.DkEwx ur /qNKcc.SWrB ZE.nylKj AA,FT /Fa-Av,1h41,32p-DQsSk-yWka RjpGS | FTRurL,DqXsskh41 |
Real-world Test Data

<table>
<thead>
<tr>
<th>Model prediction: GetSpan('[', 1, Start, Number, 1, End)</th>
<th>Const(']')</th>
<th>EOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>[CPT-101]</td>
<td>[CPT-101]</td>
<td>[CPT-101]</td>
</tr>
<tr>
<td>[CPT-101]</td>
<td>[CPT-101]</td>
<td>[CPT-101]</td>
</tr>
<tr>
<td>[CPT-11]</td>
<td>[CPT-11]</td>
<td>[CPT-11]</td>
</tr>
<tr>
<td>[CPT-1011]</td>
<td>[CPT-1011]</td>
<td>[CPT-1011]</td>
</tr>
<tr>
<td>[CPT-1011]</td>
<td>[CPT-1011]</td>
<td>[CPT-1011]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model prediction: Replace_Space_Comma(GetSpan(Proper, 1, Start, Proper, 4, End)</th>
<th>Const('.’)</th>
<th>GetLast_Prop</th>
<th>EOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jacob Ethan James</td>
<td>Jacob,Ethan,James,Alexander. -Michael</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alexander Michael</td>
<td>Elijah,Daniel,Aiden,Matthew. - Lucas</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Elijah Daniel Aiden</td>
<td>Jackson,Oliver,Jayden,Chris. - Kevin</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Matthew Lucas</td>
<td>Earth,Fire,Wind,Water. Sun</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jackson Oliver Jayden</td>
<td>Jacob,Ethan,James,Alexander. -Michael</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chris Kevin</td>
<td>Elijah,Daniel,Aiden,Matthew. - Lucas</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Earth Fire Wind Water Sun</td>
<td>Jackson,Oliver,Jayden,Chris. - Kevin</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sun</td>
<td>Earth,Fire,Wind,Water. Sun</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Encoder-Decoder Architecture

I/O Encoder

Examples

Program Decoder
I/O Encoder
Encoding Multiple I/O Examples
92% Generalization Accuracy

Program Synthesis Results

FlashfillTest

- Basic Seq.
- Att-A
- Att-B
- Att-C
- Att-C-DP
Synthesis vs Differentiable Programming

The graph compares the generalization accuracy of different methods over the number of observed IO examples. The methods compared are:

- **Synthesis, Beam=1**
- **Synthesis, Beam=100**
- **Induction**

The x-axis represents the number of observed IO examples, ranging from 1 to 4. The y-axis represents the generalization accuracy, ranging from 0% to 100%. The graph shows how each method performs with varying numbers of examples, indicating the effectiveness of Synthesis with different beam sizes compared to Induction.
Neural Program Embeddings
Graph-based Program Embedding

Miltiadis Allamanis, Marc Brockschmidt, Mahmoud Khademi. ICLR 2018
boolean f(Object target){
    for (Object elem: this.elements){
        if (elem.equals(target)){
            return true;
        }
    }
    return false;
}
Code2Seq

Uri Alon, Shaked Brody, Omer Levy, Eran Yahav. ICLR 2019
Program Execution Embedding

Ke Wang, Rishabh Singh, Zhendong Su. ICLR 2018
Program Execution Embedding

Ke Wang, Rishabh Singh, Zhendong Su. ICLR 2018
GREAT Model (Transformer + Relational Attention Bias)

```python
1 def validate_sources(sources):
2    object_name = get_content(sources, 'obj')
3    subject_name = get_content(sources, 'subj')
4    result = Result()
5    result.objects.append(object_name)
6    result.subjects.append(object_name)
7    return result
```

Vincent Hellendoorn, Charles Sutton, Rishabh Singh, Petros Maniatis, David Bieber. ICLR 2020
Human-like Programming

1. Intuition vs Enumeration

2. Improvements with Experience (Execution)

3. Multi-modal Specifications

4. Sub-problems + Compositions

5. Mistakes vs Completely correct
Karel

R. Bunel, M. Hausknecht, J. Devlin, R. Singh, P. Kohli. ICLR 2018
Karel the Robot

Input

Output

Program

Program A

def run():
    repeat(4):
        putMarker()
        move()
        turnLeft()
Karel DSL

```
Progs := def run() : s
Stmt s := while(b) : s | repeat(r) : s | s1 ; s2 | a
      | if(b) : s | ifelse(b) : s1 else : s2
Conds b := frontIsClear() | leftIsClear() | rightIsClear()
        | markersPresent() | noMarkersPresent() | not b
Actions a := move() | turnRight() | turnLeft()
            | pickMarker() | putMarker()
Costs r := 0 | 1 | ... | 19
```
Synthetic Data Generation

```python
def run():
    while (noMarkersPresent):
        putMarker()
        move()
        turnLeft()
```

Sample Random Programs & I/O Examples

Synthetic Datasets for Neural Program Synthesis. R. Shin, N. Kant, K. Gupta, C. Bender, B. Trabucco, R. Singh, D. Song ICLR 2019
Synthesis Architecture
Multiple Consistent Programs

**Input**

**Output**

---

**Program A**

```
def run():
    repeat(4):
        putMarker()
        move()
        turnLeft()
```

---

**Program B**

```
def run():
    while(noMarkersPresent):
        putMarker()
        move()
        turnLeft()
```
REINFORCE w/ Execution Feedback

1. Supervised training on synthetic dataset
2. Given I/O, Sample a program P from model
3. Execute P on I to get O’, O’ = P(I)
4. If O = O’, +1 Reward, else 0 Reward

<table>
<thead>
<tr>
<th></th>
<th>Top-1</th>
<th>Top-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervised</td>
<td>71.91</td>
<td>80.00</td>
</tr>
<tr>
<td>REINFORCE</td>
<td>71.99</td>
<td>74.11</td>
</tr>
<tr>
<td>Beam REINFORCE</td>
<td>77.68</td>
<td>82.73</td>
</tr>
</tbody>
</table>
1. Intuition vs Enumeration
2. Improvements with Experience
3. Multi-modal Specifications
4. Sub-problems + Compositions
5. Mistakes vs Completely correct
TF-Coder: Program Synthesis for Tensor Manipulations

Kensen Shi, David Bieber, Rishabh Singh arxiv 2020
Tensor Transformations in TensorFlow

Programmers need to track:

- Multiple dimensions
- Tensor shape and DType compatibility
- Mathematical correctness & Efficiency

~500 APIs

```python
import tensorflow as tf

# tf.gather
params = tf.gather(params, indices, validate_indices=None, name=None, axis=None, batch_dims=0)

# tf.one_hot
indices = tf.one_hot(indices, depth, on_value=None, off_value=None, axis=None, dtype=None, name=None)

# tf.sequence_mask
lengths = tf.sequence_mask(lengths, maxlen=None, dtype=tf.dtypes.bool, name=None)

# tf.sparse.slice
sp_input = tf.sparse.slice(sp_input, start, size, name=None)

# tf.tensordot
a, b, axes = tf.tensordot(a, b, axes, name=None)

# tf.where
condition, x=None, y=None, name=None)
```
Users ask for help on StackOverflow

Given a 1-D tensor $T$ of length $L$ which has just $N$ different values, how can I convert it to a tensor $T2$ of length $L$ with values between 0 and $N - 1$ corresponding to the values of the original tensor $T$.

Example:

$T = [45, 58, 72, 33, 45, 58, 58, 33]$  
$T2 = [0, 1, 2, 3, 0, 1, 1, 3]$  

The ordering is not important for example this also would be OK:

$T2 = [1, 0, 2, 3, 1, 0, 0, 3]$
Users ask for help on StackOverflow

Given a 1-D tensor $T$ of length $L$ which has just $N$ different values, how can I convert it to a tensor $T_2$ of length $L$ with values between 0 and $N - 1$ corresponding to the values of the original tensor $T$.

Example:

$T = [45, 58, 72, 33, 45, 58, 58, 33]$
$T_2 = [0, 1, 2, 3, 0, 1, 1, 3]$

The ordering is not important for example this also would be OK:

$T_2 = [1, 0, 2, 3, 1, 0, 0, 3]$

1 Answer

Maybe try:

- `tf.unique(T)[1]`
- `tf.unique_with_counts(T)[1]`
TFCoder Overview

Find the indices of all elements

\[
\text{in1} = [32, 53, 45, 38, 29, 89, 64, 23] \\
\text{in2} = [38, 53, 89, 38, 32, 64] \\
\text{output} = [3, 1, 5, 3, 0, 6]
\]

\[
\text{tf.cast(tf.argmax(tf.cast(tf.equal(in1, tf.expand_dims(in2, 1)), tf.int32), axis=1), tf.int32)}
\]
Two Learnt Models

I/O Examples $\rightarrow$ TF API distribution

Natural Language $\rightarrow$ TF API distribution
Natural Language Model

tf.scatter_nd

TF-IDF based cosine similarity
Neural Guided Program Compositions
Solves More Tasks + 38% Faster
Human-like Programming

1. Intuition vs Enumeration
2. Improvements with Experience
3. Multi-modal Specifications
4. Sub-problems + Compositions
5. Mistakes vs Completely correct
Latent Programmer: Discrete Latent Codes For Program Synthesis

Joey Hong, David Dohan, Rishabh Singh, Charles Sutton, Manzil Zaheer. arxiv 2020
Learning to Decompose & Compose

Problem → Decompose → Sub-problems → Compose

```python
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)

def fibonacci(n):
    if n == 0:
        return 0
    if n == 1:
        return 1
    return fibonacci(n-1) + fibonacci(n-2)
```
# Latent Programmer

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Jacob,Ethan,James 11”</td>
<td>“11:J.E.J.”</td>
</tr>
<tr>
<td>“Elijah,Daniel,Aiden 3162”</td>
<td>“3162:E.D.A”</td>
</tr>
<tr>
<td>“Rick,Oliver,Mia 26”</td>
<td>“26:R.O.M.”</td>
</tr>
<tr>
<td>“Mark,Ben,Sam 510”</td>
<td>“510:M.B.S.”</td>
</tr>
</tbody>
</table>

**LP Latent**

- TOK_14
- TOK_36
- TOK_36
- TOK_36

**LP**

- GetAll_NUMBER
- Const(():)
- getToken_ALL_CAPS_1
- Const(().)
- getToken_ALL_CAPS_2
- Const(().)
- getToken_ALL_CAPS_-1
- Const(().)
Latent Programmer Architecture

\[
\begin{array}{c|c}
I_1 & O_1 \\
\vdots & \vdots \\
I_N & O_N \\
\end{array}
\]

Latent Predictor \(lp(X)\) → Beam Search → Latent Space

Program Encoder \(ec(Y)\) → Quantize

Latent Program Decoder \(d(Z, X)\)

Beam Search → Y'
Two-level Guided Exploration

<table>
<thead>
<tr>
<th>Length</th>
<th>RobustFill Acc.</th>
<th>LP Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>94.5%</strong></td>
<td>94.0%</td>
</tr>
<tr>
<td>2</td>
<td>83.9%</td>
<td><strong>84.6%</strong></td>
</tr>
<tr>
<td>3</td>
<td><strong>72.8%</strong></td>
<td>72.2%</td>
</tr>
<tr>
<td>4</td>
<td>63.1%</td>
<td><strong>66.1%</strong></td>
</tr>
<tr>
<td>5</td>
<td>47.1%</td>
<td><strong>49.8%</strong></td>
</tr>
<tr>
<td>6</td>
<td>40.6%</td>
<td><strong>43.0%</strong></td>
</tr>
<tr>
<td>7</td>
<td>30.2%</td>
<td><strong>34.6%</strong></td>
</tr>
<tr>
<td>8</td>
<td>22.7%</td>
<td><strong>28.4%</strong></td>
</tr>
<tr>
<td>9</td>
<td>18.6%</td>
<td><strong>27.0%</strong></td>
</tr>
<tr>
<td>10</td>
<td>14.4%</td>
<td><strong>25.6%</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Latent Beam Size</th>
<th>Accuracy</th>
<th>Distinct n-Grams</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>n = 1</td>
</tr>
<tr>
<td>L = 1</td>
<td>52%</td>
<td>0.13</td>
</tr>
<tr>
<td>2</td>
<td>55%</td>
<td>0.13</td>
</tr>
<tr>
<td>3</td>
<td><strong>57%</strong></td>
<td>0.14</td>
</tr>
<tr>
<td>5</td>
<td>57%</td>
<td>0.14</td>
</tr>
<tr>
<td>10</td>
<td>56%</td>
<td><strong>0.14</strong></td>
</tr>
</tbody>
</table>
Human-like Programming

1. Intuition vs Enumeration
2. Improvements with Experience
3. Multi-modal Specifications
4. Sub-problems + Compositions
5. Mistakes vs Completely correct
Program Synthesis with a Differentiable Fixer

Matej Balog, Rishabh Singh, Petros Maniatis, Charles Sutton. arxiv 2020
## Incorrect Decoded Programs

<table>
<thead>
<tr>
<th>Input (v)</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mark Henry</td>
<td>M. Henry</td>
</tr>
<tr>
<td>Barry M. Myers</td>
<td>B. Myers</td>
</tr>
<tr>
<td>Michael Jones</td>
<td>M. Jones</td>
</tr>
<tr>
<td>Jon Sanders</td>
<td>J. Sanders</td>
</tr>
</tbody>
</table>

### RobustFill

```
Concat(SubStr(0,1), SubStr((Word, 2, Start), (Word, 2, End)))
```
# Incorrect Decoded Programs

<table>
<thead>
<tr>
<th>Input (v)</th>
<th>Output</th>
<th>Output’</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mark Henry</td>
<td>M. Henry</td>
<td>M. Henry</td>
</tr>
<tr>
<td>Barry M. Myers</td>
<td>B. Myers</td>
<td>B. M</td>
</tr>
<tr>
<td>Michael Jones</td>
<td>M. Jones</td>
<td>M. Jones</td>
</tr>
<tr>
<td>Jon Sanders</td>
<td>J. Sanders</td>
<td>J. Sanders</td>
</tr>
</tbody>
</table>

RobustFill

\[ \text{Concat(SubStr(0,1), SubStr((Word, 2, Start), (Word, 2, End)))} \]
# Beam Search over Program Distribution

<table>
<thead>
<tr>
<th>Input (v)</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mark Henry</td>
<td>M. Henry</td>
</tr>
<tr>
<td>Barry M. Myers</td>
<td>B. Myers</td>
</tr>
<tr>
<td>Michael Jones</td>
<td>M. Jones</td>
</tr>
<tr>
<td>Jon Sanders</td>
<td>J. Sanders</td>
</tr>
</tbody>
</table>

RobustFill

\[ P_1 \quad P_2 \quad P_3 \quad P_4 \quad \ldots \quad P_{10} \]
# Learning to Fix Mistakes

<table>
<thead>
<tr>
<th>Input (v)</th>
<th>Output</th>
<th>Output’</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mark Henry</td>
<td>M. Henry</td>
<td>M. Henry</td>
</tr>
<tr>
<td>Barry M. Myers</td>
<td>B. Myers</td>
<td>B. M</td>
</tr>
<tr>
<td>Michael Jones</td>
<td>M. Jones</td>
<td>M. Jones</td>
</tr>
<tr>
<td>Jon Sanders</td>
<td>J. Sanders</td>
<td>J. Sanders</td>
</tr>
</tbody>
</table>

## Differentiable Fixer

```
Concat(SubStr(0,1), SubStr((Word, -1, Start), (Word, -1, End)))
```
Synthesis with Differentiable Fixer

\[
\text{Program Decoder} \quad \quad \arg\max \text{ LSTM}(z_E)
\]

\[
\text{Example Encoder} \quad \quad \maxpool_{1 \leq n \leq N} f_{iO}(i_n, o_n)
\]

Latent Space

\[
z_E
\]
Synthesis with Differentiable Fixer

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$H$</th>
<th>Params</th>
<th>Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beam</td>
<td>512</td>
<td>2.2$M$</td>
<td>53%</td>
</tr>
<tr>
<td>Beam</td>
<td>576</td>
<td>2.9$M$</td>
<td>54%</td>
</tr>
<tr>
<td>Beam</td>
<td>608</td>
<td>3.2$M$</td>
<td>56%</td>
</tr>
<tr>
<td>Fixer</td>
<td>512</td>
<td>3.0$M$</td>
<td>61%</td>
</tr>
</tbody>
</table>
Human-like Program Synthesis

1. Intuition vs Enumeration
   RobustFill [ICLR 2017, ICML 2017]

2. Improvements with Experience
   Karel [NeurIPS 2017, ICLR 2018]

3. Multi-modal Specifications
   TF-Coder [arxiv 2020]

4. Sub-problems + Compositions
   Latent Programmer [2020], Bustle [ICLR 2021]

5. Mistakes vs Completely correct
   Differentiable Fixer [2020]
Human-like Program Synthesis

1. Intuition vs Enumeration
   - RobustFill [ICLR 2017, ICML 2017]

2. Improvements with Experience
   - Karel [NeurIPS 2017, ICLR 2018]

3. Multi-modal Specifications
   - TF-Coder [arxiv 2020]

4. Sub-problems + Compositions
   - Latent Programmer [2020], Bustle [ICLR 2021]

5. Mistakes vs Completely correct
   - Differentiable Fixer [2020]

- New Neural Architectures
- Differentiable Reasoning
- Learning w Rich Feedback
- General Real-world Programs