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

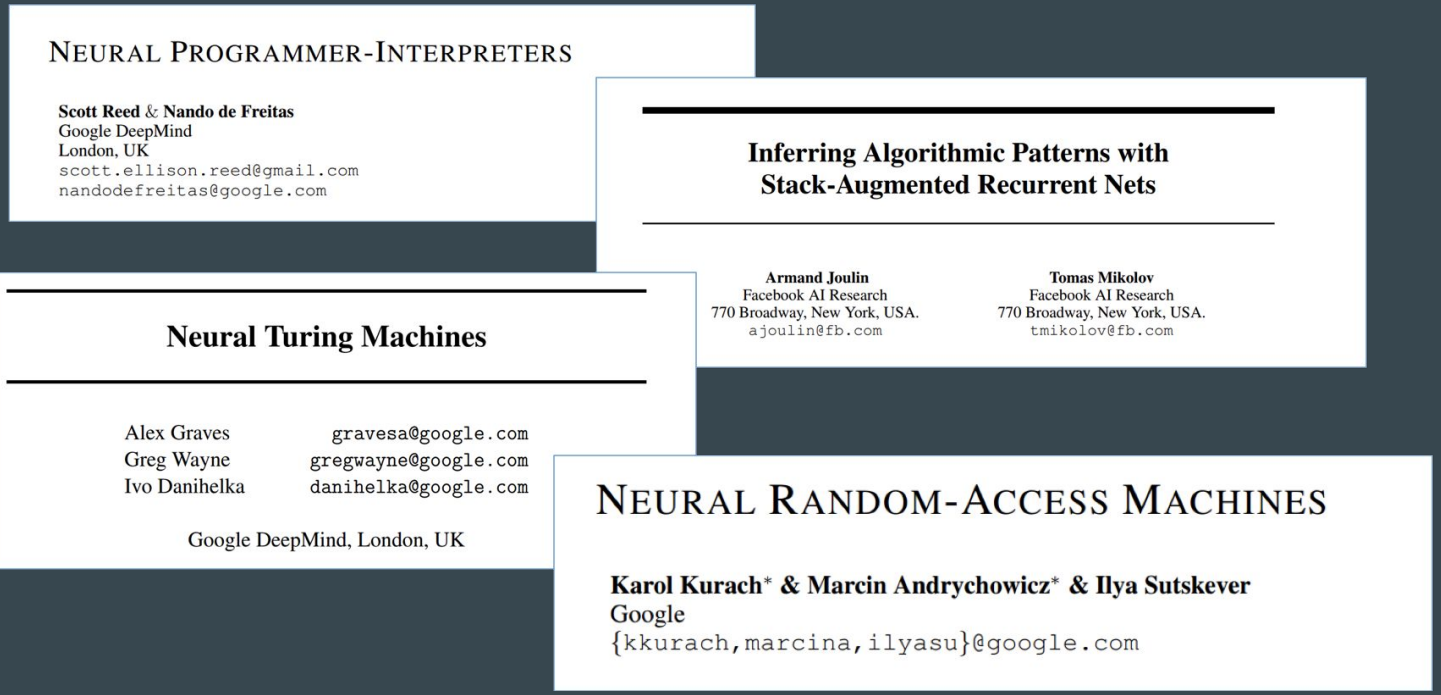
Spec

I/O Examples
Natural Language
Partial Programs

```
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)
```

Logic Basics Experience Samples

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

Inferring Algorithmic Patterns with Stack-Augmented Recurrent Nets

Armand Joulin
Facebook AI Research
770 Broadway, New York, USA.
ajoulin@fb.com

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

Neural Turing Machines

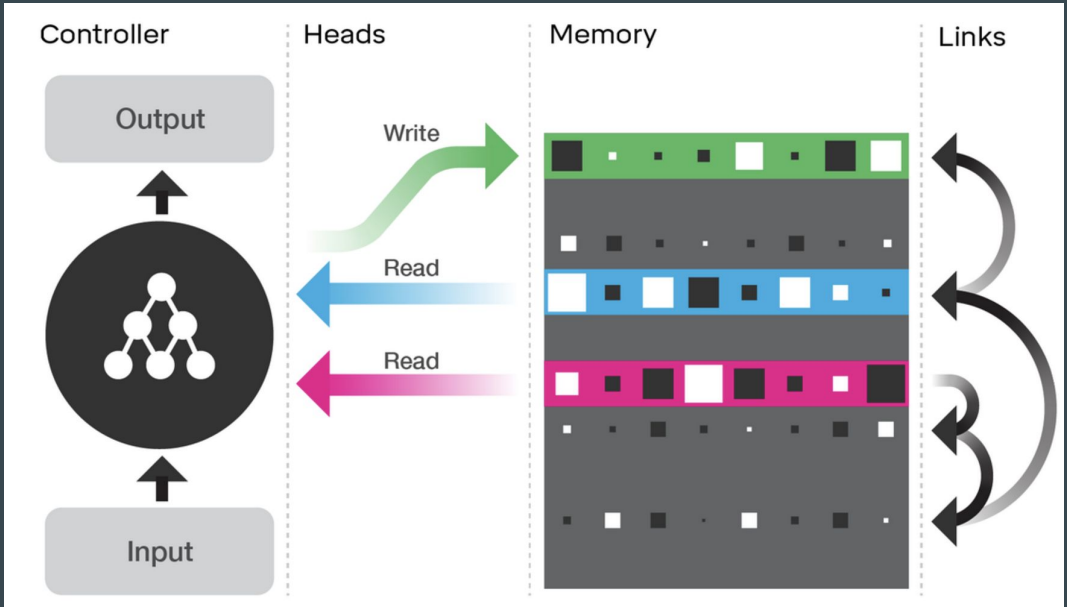
Alex Graves gravesa@google.com
Greg Wayne gregwayne@google.com
Ivo Danihelka danihelka@google.com

Google DeepMind, London, UK

NEURAL RANDOM-ACCESS MACHINES

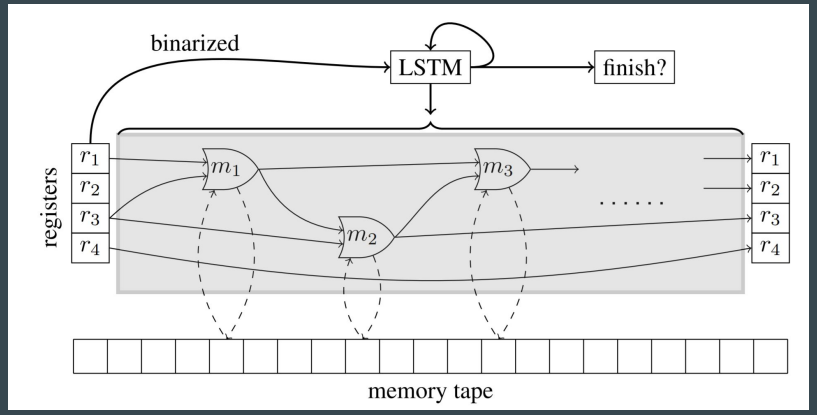
Karol Kurach* **& Marcin Andrychowicz*** **& Ilya Sutskever**
Google
{kkurach,marcina,ilyasu}@google.com

Differentiable Neural Computer



Graves et al. Nature 2016

Neural Random Access Machines



$$\begin{aligned}\mathrm{ADD}(x,y) &= x+y\\ \mathrm{MAX}(x,y) &= \max(x,y)\\ \mathrm{EQZ}(x) &= \begin{cases}1 & \text{if } x=0\\ 0 & \text{otherwise}\end{cases}\\ \mathrm{ZERO} &= 0\end{aligned}$$

$$m_{\mathrm{ADD}}(\vec{A},\vec{B})_\ell = \sum_{0\le j,k<M} A_j B_k [j+k=\ell]$$

$$2 \to \begin{bmatrix}0\\0\\1\\0\\\vdots\\0\end{bmatrix} \in \mathbb{Z}_M \qquad m_{\mathrm{ZERO}} = \begin{bmatrix}1\\0\\0\\\vdots\\0\end{bmatrix} \qquad m_{\mathrm{EQZ}}(\vec{A}) = \begin{bmatrix}A_0\\1-A_0\\0\\\vdots\\0\end{bmatrix}$$

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

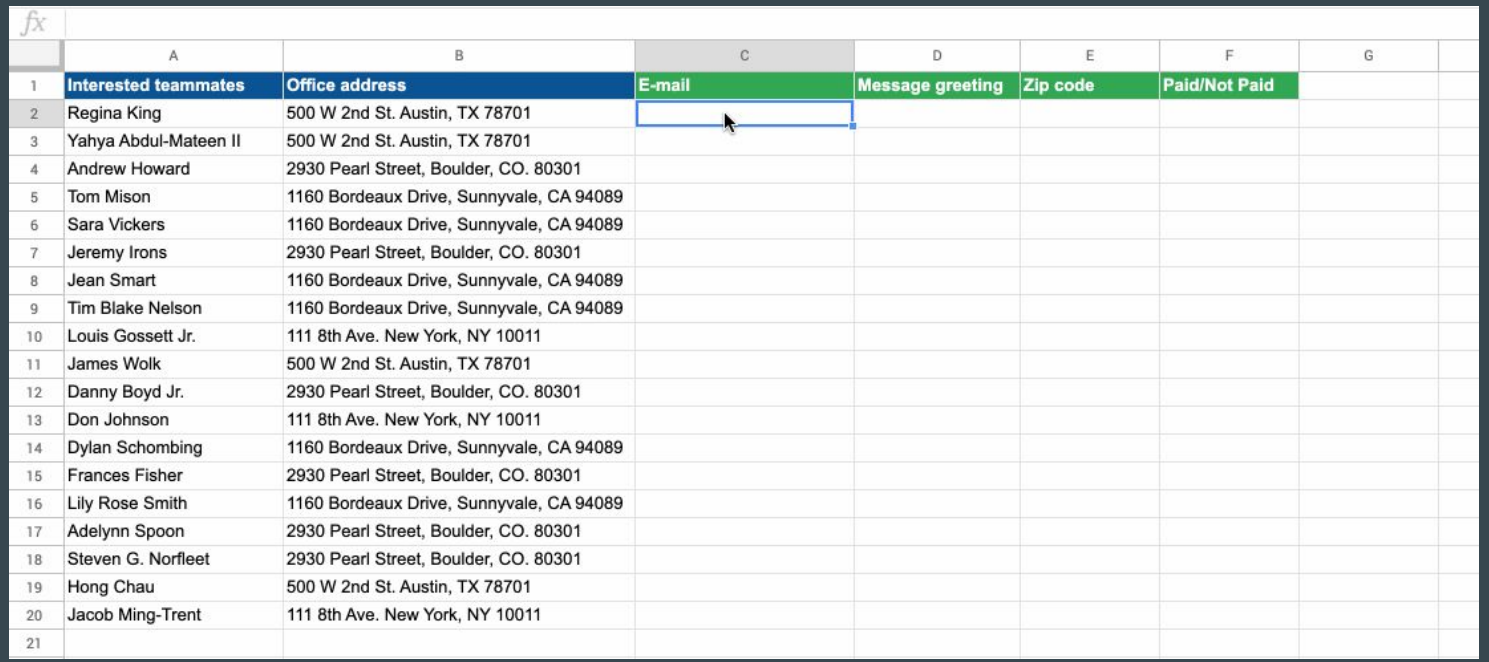
1. Intuition vs Enumeration

No Super-human computation!

2. Improvements with Experience
3. Multi-modal Specifications
4. Sub-problems + Compositions
5. Mistakes vs Completely correct

SmartFill

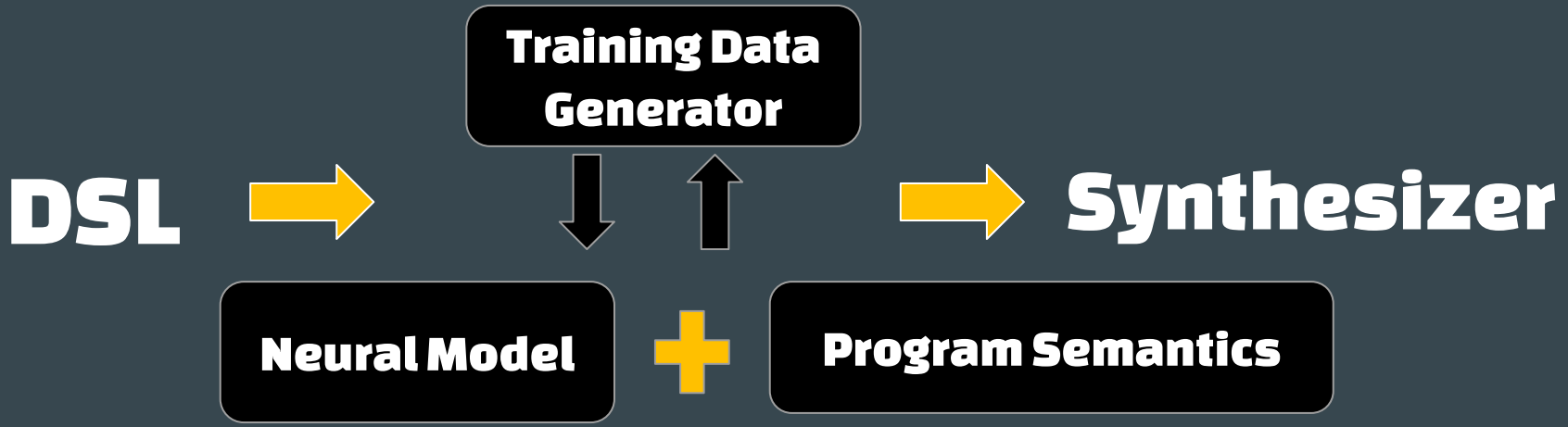
SmartFill



	A	B	C	D	E	F	G
1	Interested teammates	Office address	E-mail	Message greeting	Zip code	Paid/Not Paid	
2	Regina King	500 W 2nd St. Austin, TX 78701					
3	Yahya Abdul-Mateen II	500 W 2nd St. Austin, TX 78701					
4	Andrew Howard	2930 Pearl Street, Boulder, CO. 80301					
5	Tom Mison	1160 Bordeaux Drive, Sunnyvale, CA 94089					
6	Sara Vickers	1160 Bordeaux Drive, Sunnyvale, CA 94089					
7	Jeremy Irons	2930 Pearl Street, Boulder, CO. 80301					
8	Jean Smart	1160 Bordeaux Drive, Sunnyvale, CA 94089					
9	Tim Blake Nelson	1160 Bordeaux Drive, Sunnyvale, CA 94089					
10	Louis Gossett Jr.	111 8th Ave. New York, NY 10011					
11	James Wolk	500 W 2nd St. Austin, TX 78701					
12	Danny Boyd Jr.	2930 Pearl Street, Boulder, CO. 80301					
13	Don Johnson	111 8th Ave. New York, NY 10011					
14	Dylan Schombing	1160 Bordeaux Drive, Sunnyvale, CA 94089					
15	Frances Fisher	2930 Pearl Street, Boulder, CO. 80301					
16	Lily Rose Smith	1160 Bordeaux Drive, Sunnyvale, CA 94089					
17	Adelynn Spoon	2930 Pearl Street, Boulder, CO. 80301					
18	Steven G. Norfleet	2930 Pearl Street, Boulder, CO. 80301					
19	Hong Chau	500 W 2nd St. Austin, TX 78701					
20	Jacob Ming-Trent	111 8th Ave. New York, NY 10011					
21							

Google Sheets will soon be able to autocomplete data for you

Neuro-symbolic Program Synthesis



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

Synthetic Dataset Generation

Reference program: GetToken_Alphanum_3 | GetFrom_Colon_1 | GetFirst_Char_4	
Ud 9:25,JV3 Obb	2525,JV3 bbUd92
zLny xmHg 8:43 A44q	843 A44qzLny
A6 g45P 10:63 Jf	1063 JfA6g4
cuL.zF.dDX,12:31	dDX31cuLz
ZiG OE bj3u 7:11	bj3u11ZiGO

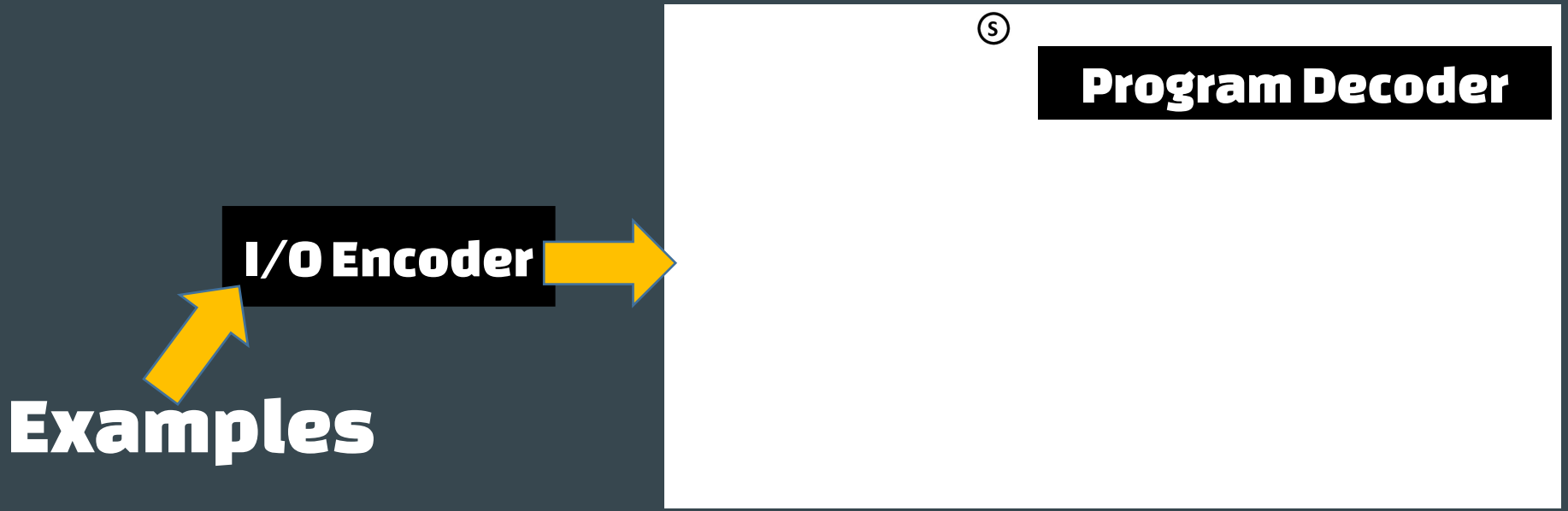
Reference program: GetToken_WS_-1(GetSpan(Number, 1 End, '/', 3, End)) | Const('R') | GetToken_Word_5 | Const('L') | Const(',') | ToProper(GetToken_Word_3) | GetToken_Alphanum_5 | EOS	
aC Ic 3.rFL JiW.MmB fzYoa TX oNpV fHm /ai WHGM Pgso.OXp VKW Jo R9 0JUF / / Xir	JUF RMmBL,RVKW
wa.Xvq-wo-isxn KD.qxpkH mACHu/ZNI Qhs-DAr,UAr-UcP.Ps xjK-JL0,AB.tdn,1-fyA//eZ	fyARKDL,TdnAB
Iceg gbeOz ck CbwoZ /Zmfb WMyoO /10 CQlXs,EkeFJAxi Ld a9z aSd Cse9 Ey xAG /QVqq njc ukx	Ey xAG RZmfbL,QvqqEy
qm/CsPc oaSUW,wKz.rRH,jFqO.PGihT IE-2,NL zzToV-2W6z,dE,Ptl /dSZR.Xel/xyEA-qN kf.Yo	XelRrRHL,QnXel
wUx -7.ND7.xiE.DkEwx ur /qNKCc.SWrB ZE.nylKj AA,FT/ /Fa-Av,lh4l,32p-DQsSk-yWka RjpGS	FTRurL,DqXsskh4l

Real-world Test Data

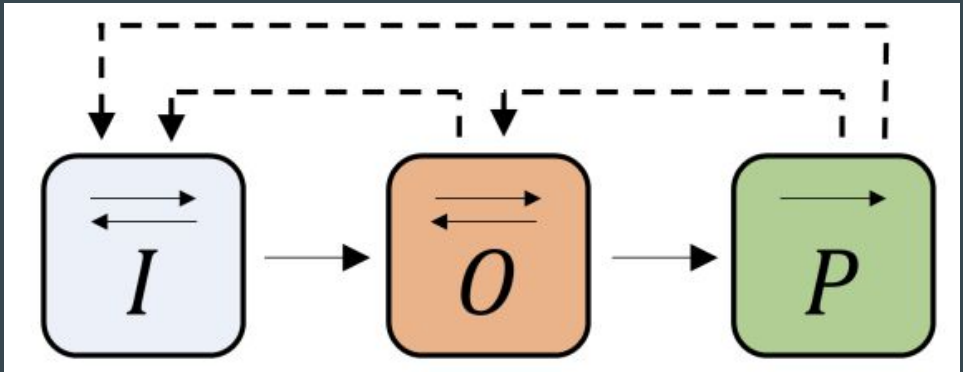
Model prediction: GetSpan(‘[’, 1, Start, Number, 1, End) | Const(‘]’) | EOS		
[CPT-101	[CPT-101]	[CPT-101]
[CPT-101	[CPT-101]	[CPT-101]
[CPT-11]	[CPT-11]	[CPT-11]
[CPT-1011]	[CPT-1011]	[CPT-1011]
[CPT-1011	[CPT-1011]	[CPT-1011]

Model prediction: Replace_Space_Comma(GetSpan(Proper, 1, Start, Proper, 4, End) | Const(‘.’) | GetLast_Proper | EOS		
Jacob Ethan James Alexander Michael	Jacob,Ethan,James,Alexander.-Michael	Jacob,Ethan,James,Alexander.-Michael
Elijah Daniel Aiden Matthew Lucas	Elijah,Daniel,Aiden,Matthew.-Lucas	Elijah,Daniel,Aiden,Matthew.-Lucas
Jackson Oliver Jayden Chris Kevin	Jackson,Oliver,Jayden,Chris.-Kevin	Jackson,Oliver,Jayden,Chris.-Kevin
Earth Fire Wind Water Sun	Earth,Fire,Wind,Water.Sun	Earth,Fire,Wind,Water.Sun

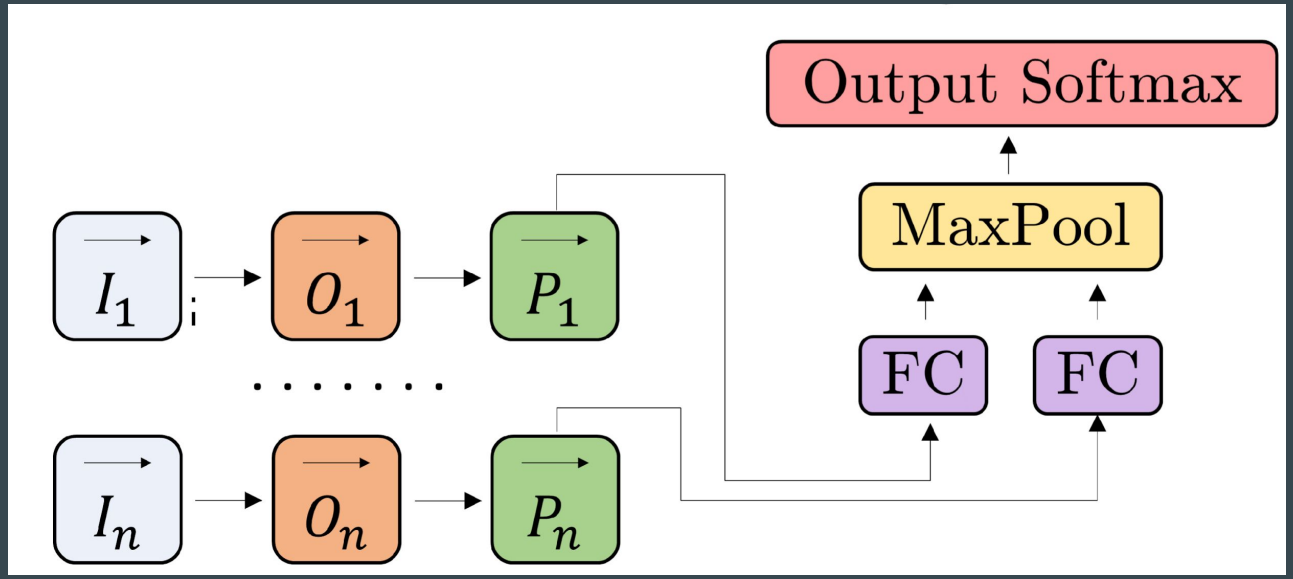
Encoder-Decoder Architecture



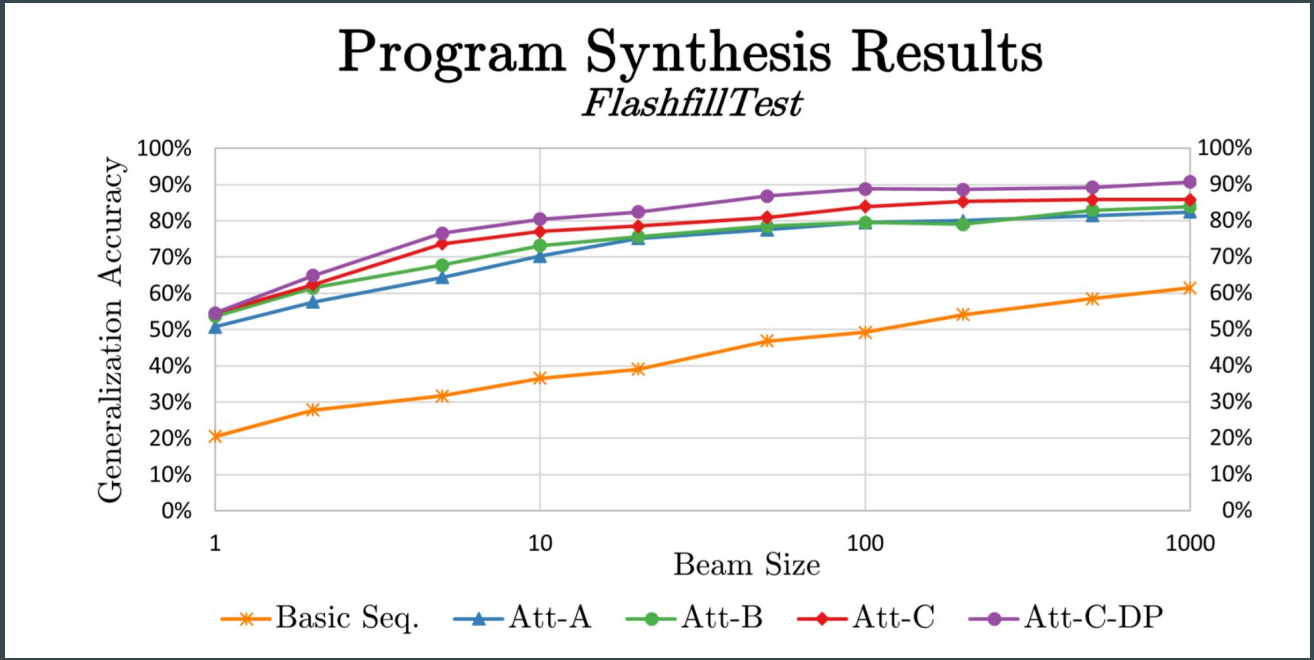
I/O Encoder



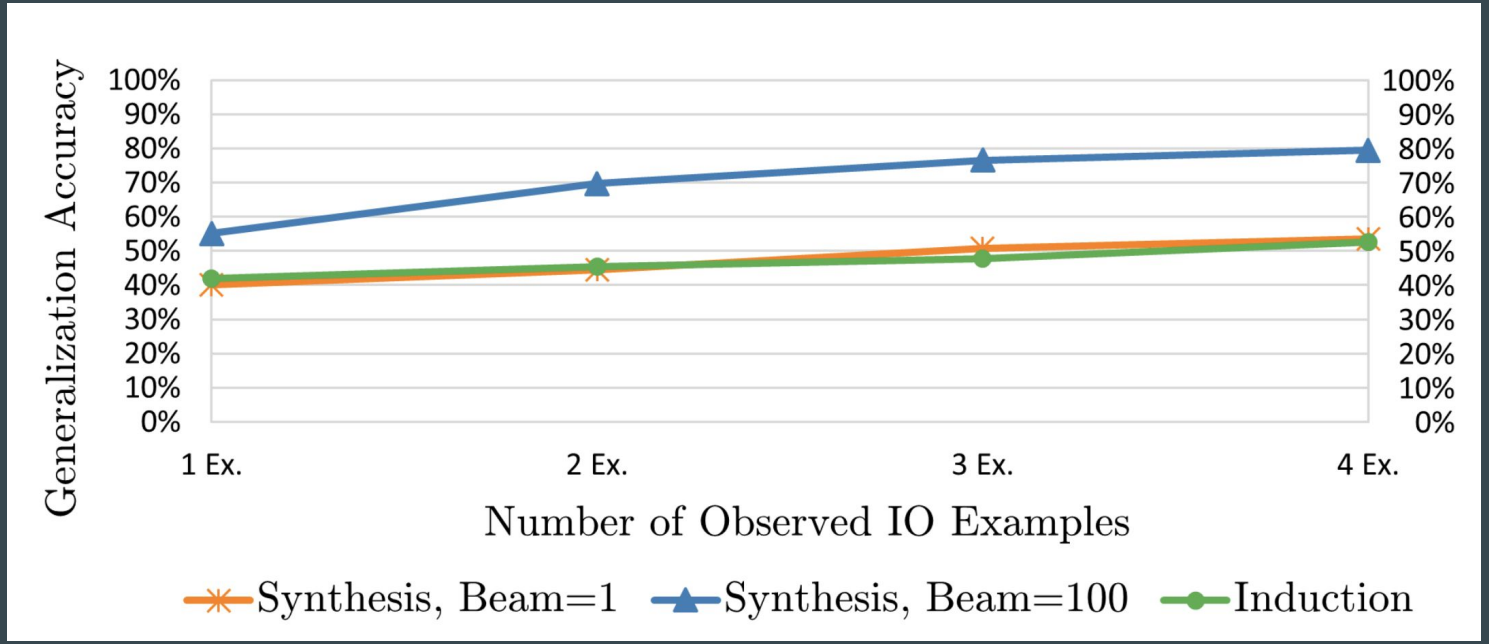
Encoding Multiple I/O Examples



92% Generalization Accuracy



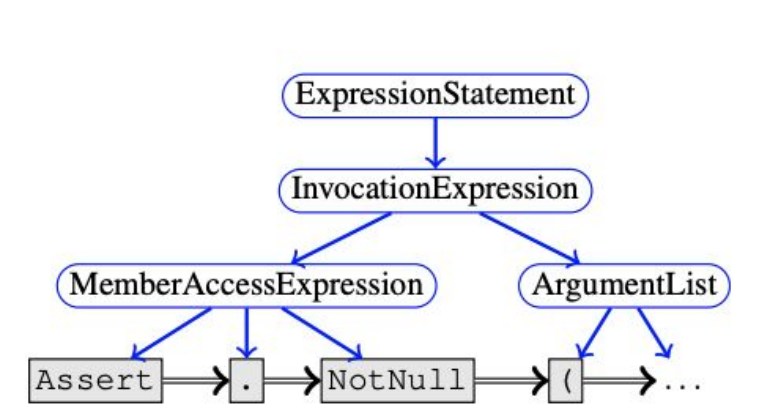
Synthesis vs Differentiable Programming

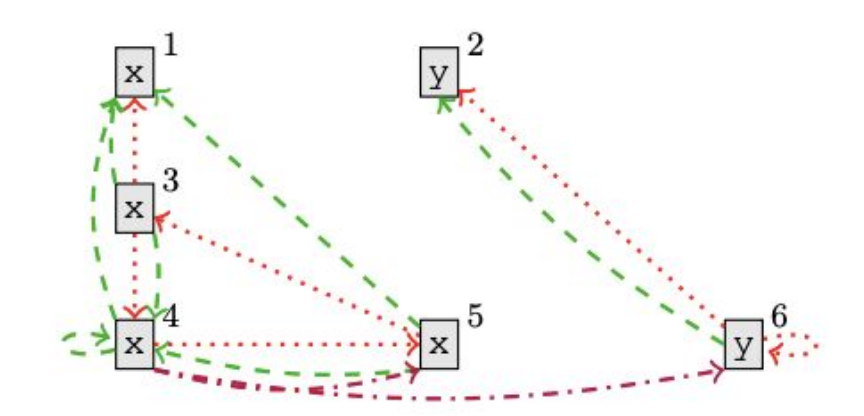


Neural Program Embeddings

Graph-based Program Embedding

```
var clazz=classTypes["Root"].Single() as JsonCodeGenerator.ClassType;
Assert.NotNull(clazz);

var first=classTypes["RecClass"].Single() as JsonCodeGenerator.ClassType;
Assert.NotNull( clazz );

Assert.Equal("string", first.Properties["Name"].Name);
Assert.False(clazz.Properties["Name"].IsArray);
```

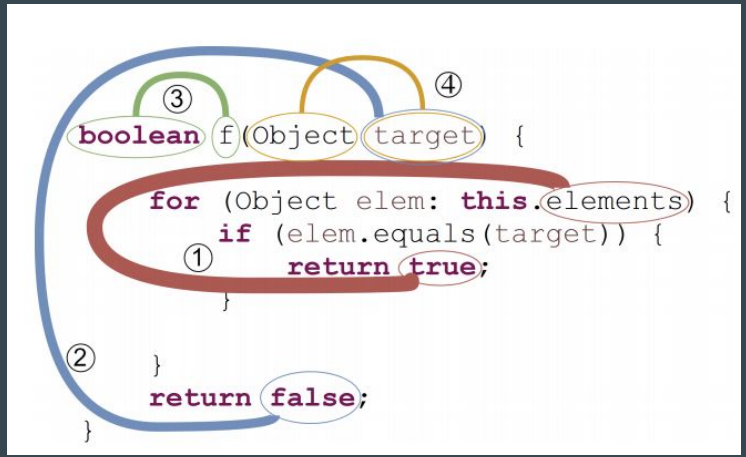




Miltiadis Allamanis, Marc Brockschmidt, Mahmoud Khademi. ICLR 2018

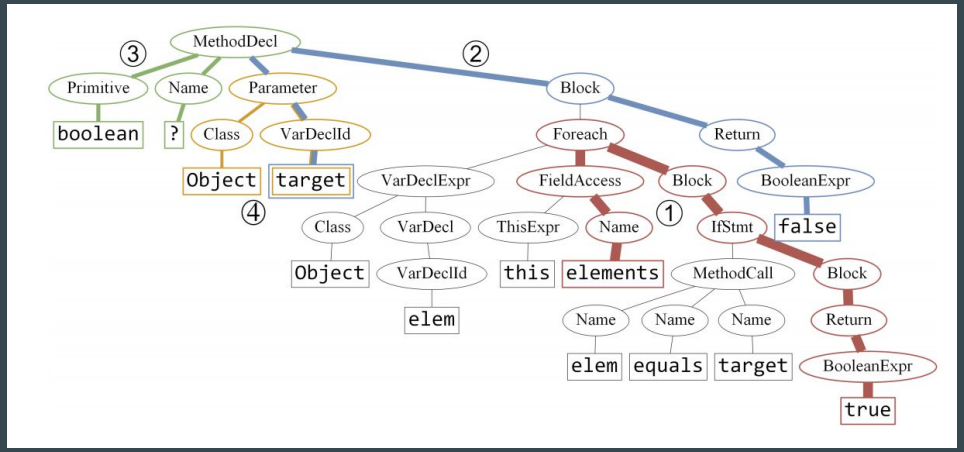
Code2vec

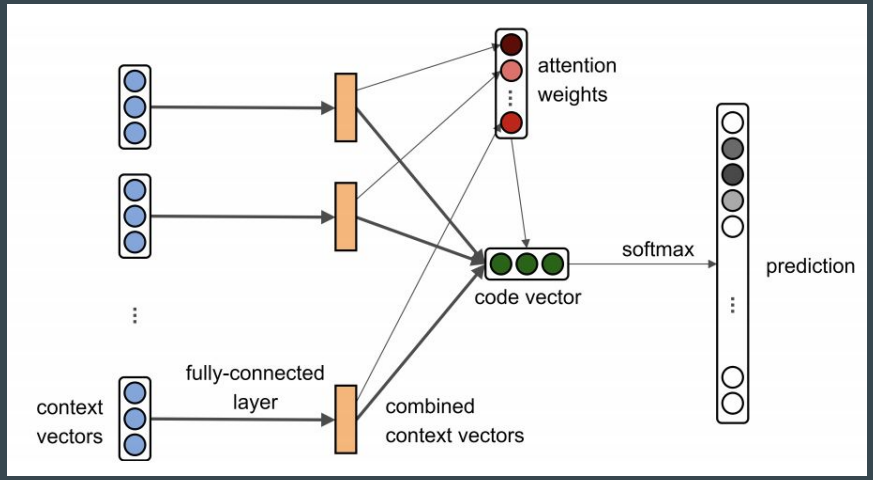
```
boolean f(Object target){
    for (Object elem: this.elements){
        if(elem.equals(target)){
            return true;
        }
    }
    return false;
}
```



Uri Alon, Meital Zilberstein, Omer Levy, Eran Yahav. POPL 2019

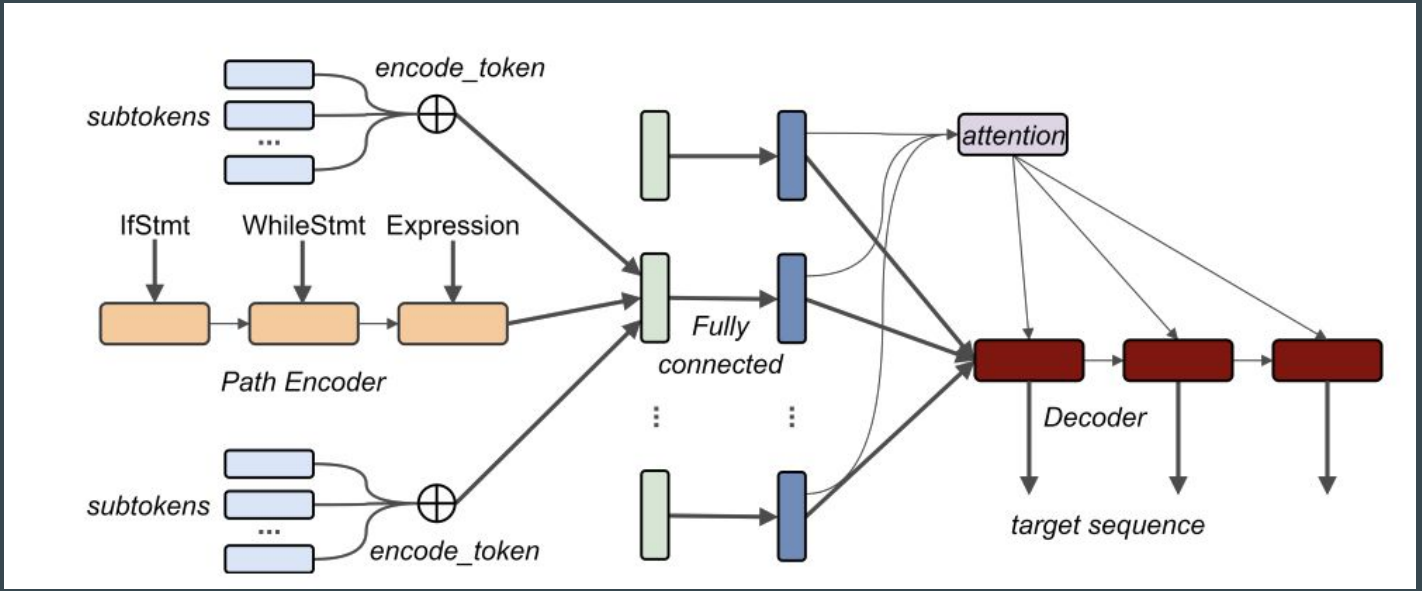
Code2vec





Uri Alon, Meital Zilberstein, Omer Levy, Eran Yahav. POPL 2019

Code2Seq



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

Program Execution Embedding

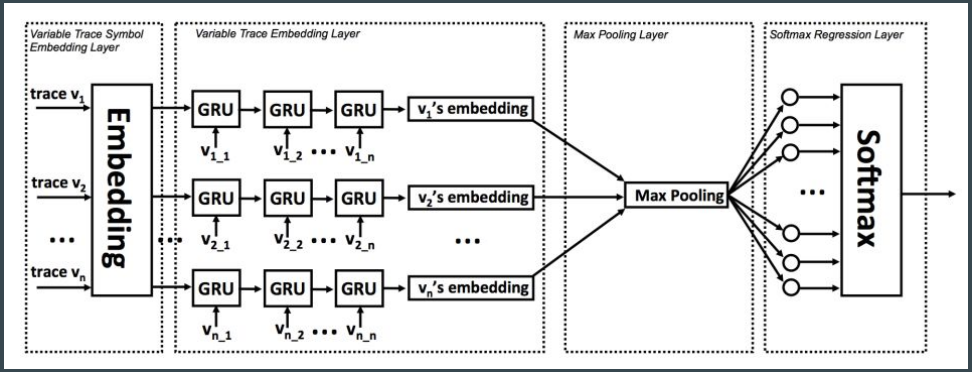
```
static int[] BubbleSort(int[] A) {
    int left = 0;
    int right = A.Length - 1;

    for (int i = right;i > left;i--) {
        for (int j = left;j < i;j++) {
            if (A[j] > A[j + 1]) {
                int tmp = A[j];

                A[j] = A[j + 1];
                // instrumentation line
                Console.WriteLine(
                    string.Join(",", A)
                );

                A[j + 1] = tmp;
                // instrumentation line
                Console.WriteLine(
                    string.Join(",", A)
                );
    }}}

    return A;
}
```

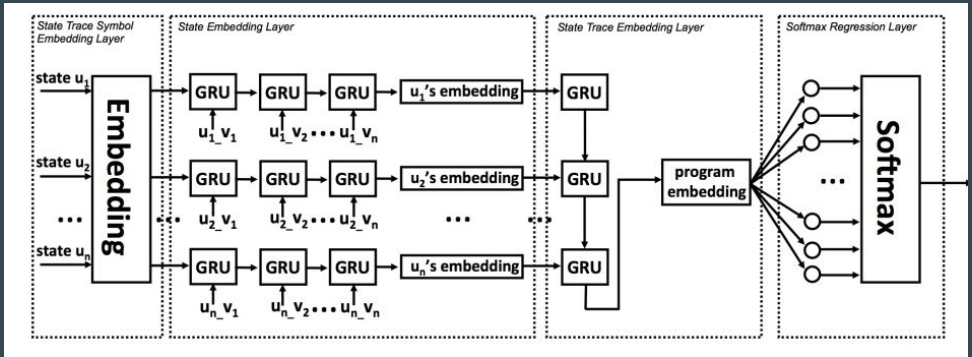
```
static int[] InsertionSort(int[] A) {
    int left = 0;
    int right = A.Length;

    for (int i = left;i < right;i++) {
        for (int j = i - 1;j >= left;j--) {
            if (A[j] > A[j + 1]) {
                int tmp = A[j];

                A[j] = A[j + 1];
                // instrumentation line
                Console.WriteLine(
                    string.Join(",", A)
                );

                A[j + 1] = tmp;
                // instrumentation line
                Console.WriteLine(
                    string.Join(",", A)
                );
    }}}

    return A;
}
```

Bubble	Insertion
[5,5,1,4,3]	[5,5,1,4,3]
[5,8,1,4,3]	[5,8,1,4,3]
[5,1,1,4,3]	[5,1,1,4,3]
[5,1,8,4,3]	[5,1,8,4,3]
[1,1,8,4,3]	[5,1,4,4,3]
[1,5,8,4,3]	[5,1,4,8,3]
[1,5,4,4,3]	[5,1,4,3,3]
[1,5,4,8,3]	[5,1,4,3,8]
[1,4,4,8,3]	[1,1,4,3,8]
[1,4,5,8,3]	[1,5,4,3,8]
[1,4,5,3,3]	[1,4,4,3,8]
[1,4,5,3,8]	[1,4,5,3,8]
[1,4,3,3,8]	[1,4,3,3,8]
[1,4,3,5,8]	[1,4,3,5,8]
[1,3,3,5,8]	[1,3,3,5,8]
[1,3,4,5,8]	[1,3,4,5,8]

Ke Wang, Rishabh Singh, Zhendong Su. ICLR 2018

Program Execution Embedding

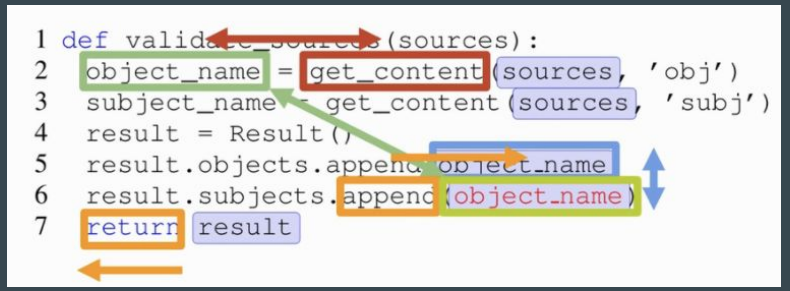




Variable Trace	State Trace
$\{max\_val : -\infty\}$	$\{max\_val : -\infty, item : \perp\}$
$\{item : 1\}$	$\{max\_val : -\infty, item : 1\}$
$\{max\_val : 1\}$	$\{max\_val : 1, item : 1\}$
$\{item : 5\}$	$\{max\_val : 1, item : 5\}$
$\{max\_val : 5\}$	$\{max\_val : 5, item : 5\}$
$\{item : 3\}$	$\{max\_val : 5, item : 3\}$

Ke Wang, Rishabh Singh, Zhendong Su. ICLR 2018

GREAT Model (Transformer+Relational Attention Bias)




```
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
```

$$\alpha_{ij} \sim \mathbf{q}_i \mathbf{k}_j^\top / \sqrt{N}$$

$$\alpha_{ij} \sim (\mathbf{q}_i + b_{ij}) \mathbf{k}_j^\top / \sqrt{N}$$

$$b_{ij} = \begin{cases} W_\tau \mathbf{e}_\tau, & \exists\, edge(i,j,\tau) \\ 0, & \text{otherwise} \end{cases}$$

Vincent Hellendoorn, Charles Sutton, Rishabh Singh, Petros Maniatis, David Bieber. ICLR 2020

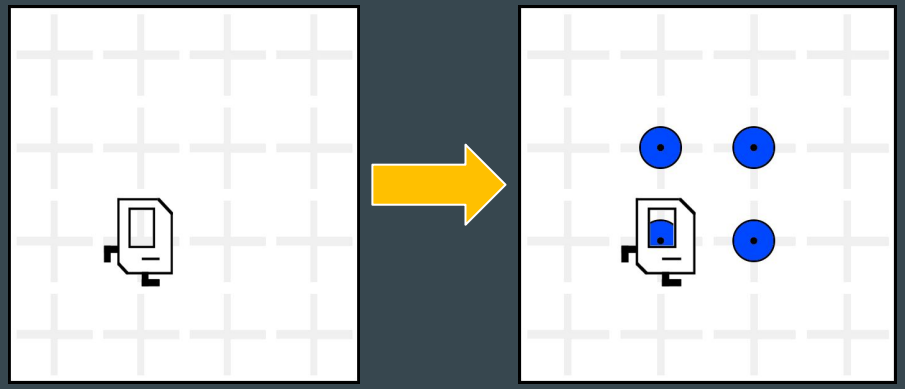
Human-like Programming

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 A

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

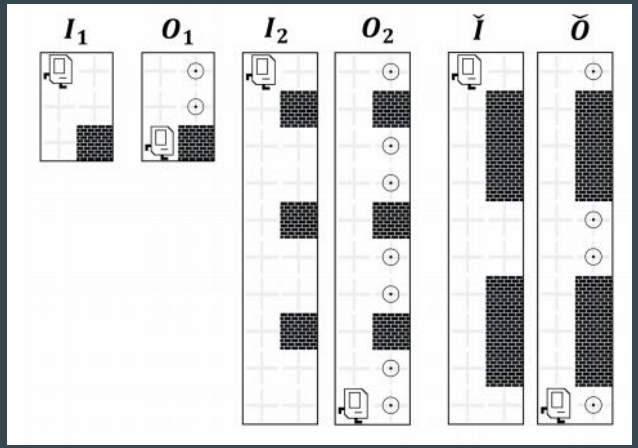
Program

Karel DSL

$$
\begin{array}{rcl}
\text{Prog } p & := & \texttt{def run() : } s \\
\text{Stmt } s & := & \texttt{while}(b) : s \mid \texttt{repeat}(r) : s \mid s_1; s_2 \mid a \\
 & \mid & \texttt{if}(b) : s \mid \texttt{ifelse}(b) : s_1 \texttt{ else} : s_2 \\
\text{Cond } b & := & \texttt{frontIsClear}() \mid \texttt{leftIsClear}() \mid \texttt{rightIsClear}() \\
 & \mid & \texttt{markersPresent}() \mid \texttt{noMarkersPresent}() \mid \texttt{not } b \\
\text{Action a} & := & \texttt{move}() \mid \texttt{turnRight}() \mid \texttt{turnLeft}() \\
 & \mid & \texttt{pickMarker}() \mid \texttt{putMarker}() \\
\text{Cste } r & := & 0 \mid 1 \mid \cdots \mid 19
\end{array}
$$

Synthetic Data Generation

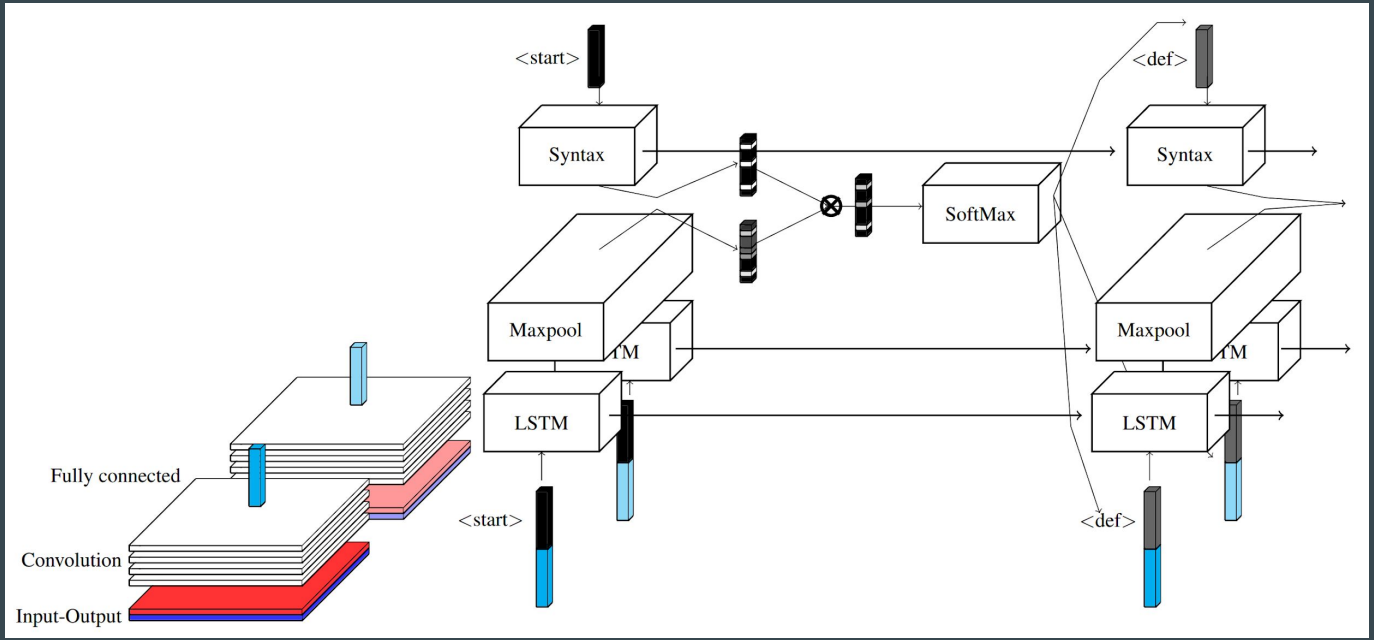
```
def run():
  while(noMarkersPresent):
    putMarker()
    move()
    turnLeft()
```



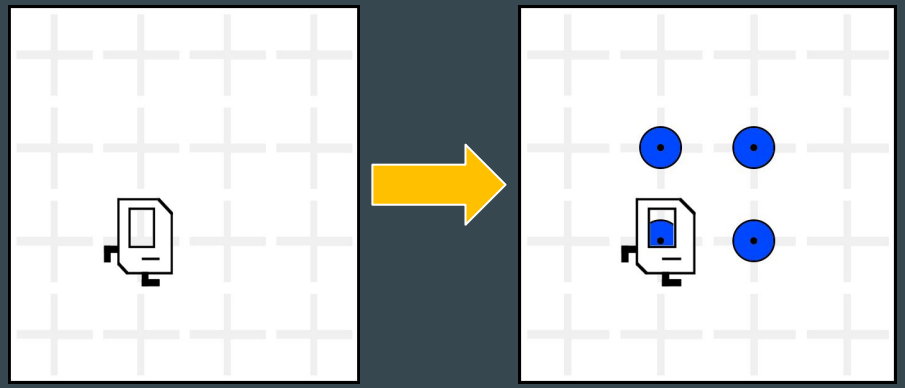
Sample Random Programs & I/O Examples

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

	Top-1	Top-5
Supervised	71.91	80.00
REINFORCE	71.99	74.11
Beam REINFORCE	**77.68**	**82.73**

Human-like Programming

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

Programmers need to track

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

~500 APIs

tf.gather

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

tf.one_hot

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

tf.sequence_mask

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

tf.sparse.slice

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

tf.tensordot

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

tf.where

```
tf.where(
    condition,
    x=None,
    y=None,
    name=None
)
```

Users ask for help on StackOverflow

1

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]
```

https://stackoverflow.com/questions/53054668/

Users ask for help on StackOverflow

1

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]
```

1 Answer

1

Maybe try:

tf.unique(T)[1]

tf.unique_with_counts(T)[1]

share improve this answer follow

https://stackoverflow.com/questions/53054668/

TFCoder Overview

Find the indices of all elements

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

TFCoder

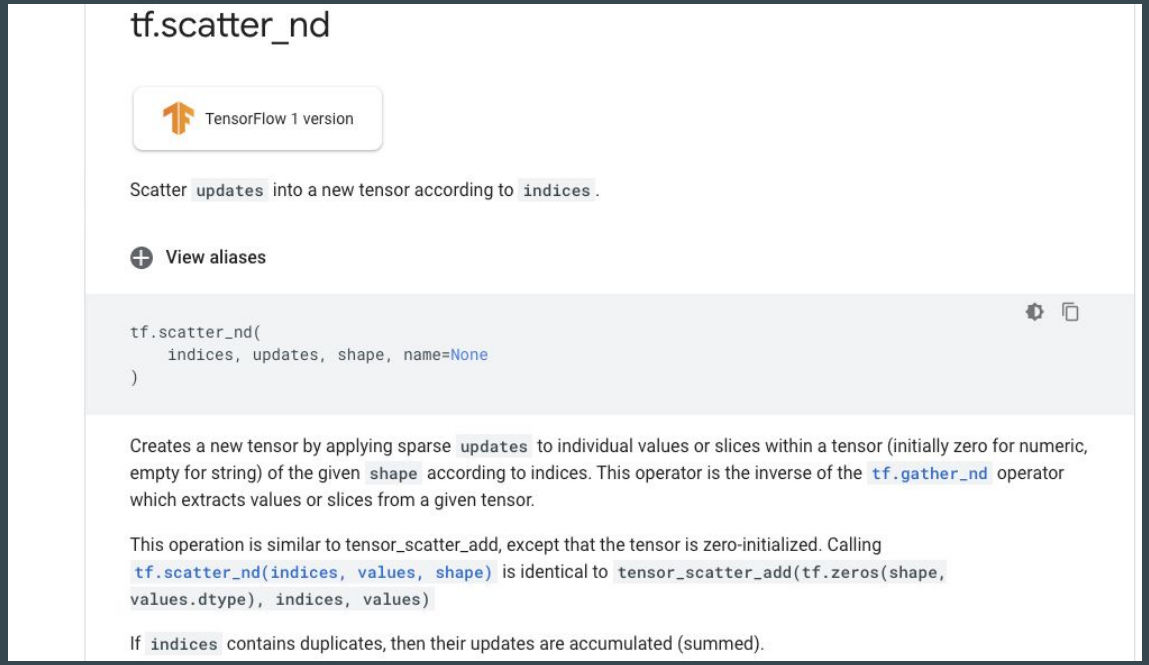
```
tf.cast(tf.argmax(tf.cast(tf.equal(in1, tf.expand_dims(
    in2, 1)), tf.int32), axis=1), tf.int32)
```

Two Learnt Models

I/O Examples → TF API distribution

Natural Language → TF API distribution

Natural Language Model



tf.scatter_nd

TensorFlow 1 version

Scatter `updates` into a new tensor according to `indices`.

View aliases

```
tf.scatter_nd(
    indices, updates, shape, name=None
)
```

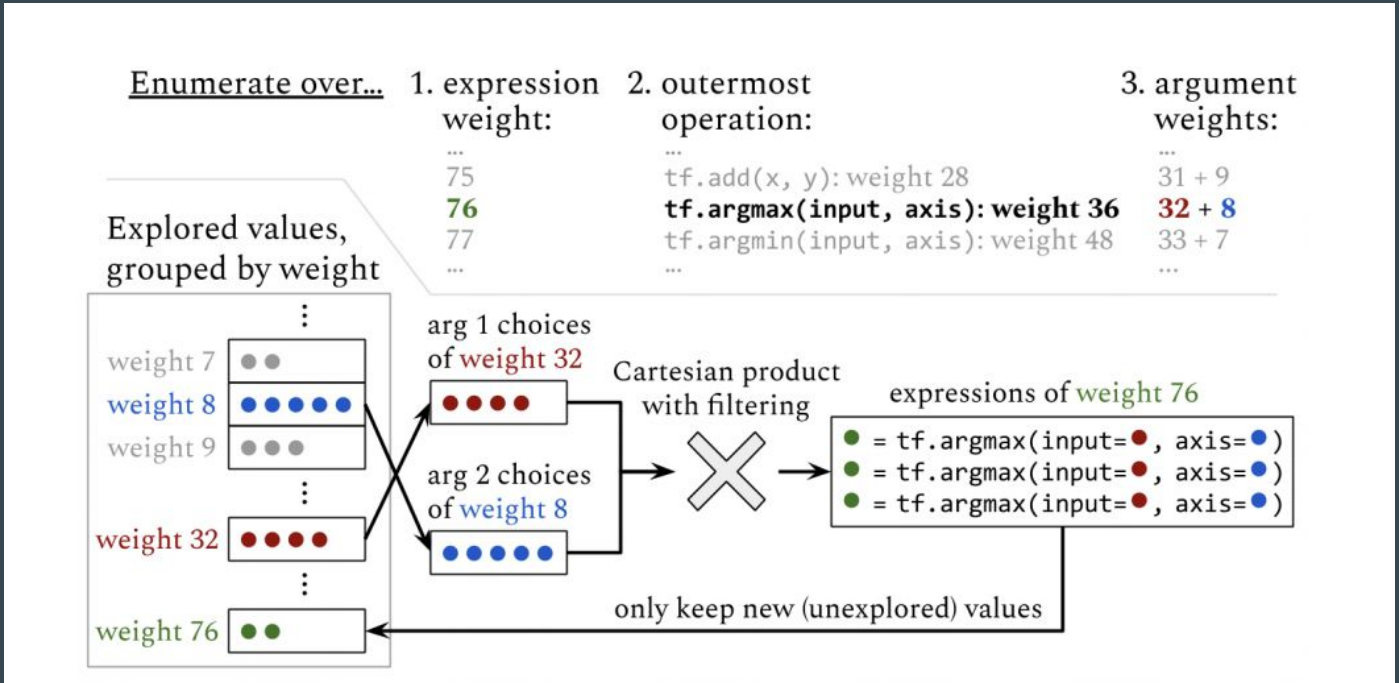
Creates a new tensor by applying sparse `updates` to individual values or slices within a tensor (initially zero for numeric, empty for string) of the given `shape` according to indices. This operator is the inverse of the `tf.gather_nd` operator which extracts values or slices from a given tensor.

This operation is similar to tensor_scatter_add, except that the tensor is zero-initialized. Calling `tf.scatter_nd(indices, values, shape)` is identical to `tensor_scatter_add(tf.zeros(shape, values.dtype), indices, values)`

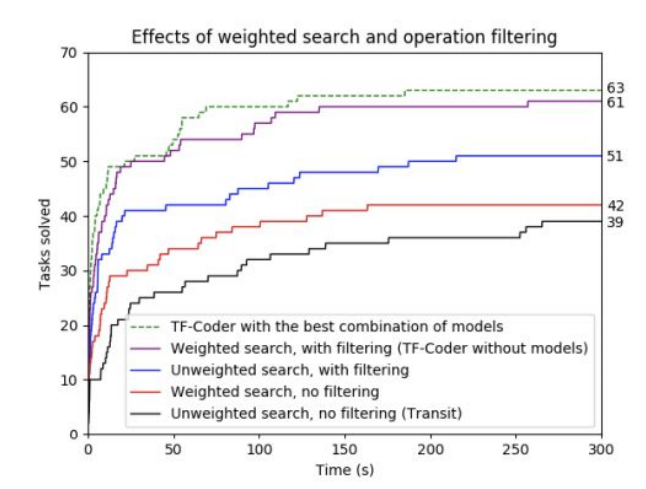
If `indices` contains duplicates, then their updates are accumulated (summed).

TF-IDF based cosine similarity

Neural Guided Program Compositions



Solves More Tasks + 38% Faster



Model	Tasks solved	Num faster (avg. speedup)	Num slower (avg. speedup)	Time for 61 tasks (s)	Total speedup	Avg. speedup
TF-Coder without any models	61	–	–	1261.1	–	–
(A) CE, *weight* = $w_i^{\max}$	62	33 (44.1%)	15 (−25.1%)	979.2	22.4%	20.1%
(B) F_1, *weight* = $w_i^{\max}$	**63**	38 (45.3%)	14 (−28.0%)	**866.1**	**31.3%**	24.9%
(C) F_2, *weight* = $w_i^{\max}$	**63**	**41** (46.3%)	**7** (−85.7%)	1045.3	17.1%	**25.0%**
(X) Naïve Bayes, $k = 3$, $p = 0.5$	61	**27** (39.8%)	13 (−16.4%)	**1055.7**	**16.3%**	12.2%
(Y) Naïve Bayes, $k = 10$, $p = 0.75$	61	26 (40.5%)	7 (−17.3%)	1071.9	15.0%	13.5%
(Z) TF-IDF, $k = 5$, *minScore* = 0.15	61	23 (43.3%)	**7** (−12.5%)	1167.6	7.4%	**16.1%**
(B) with **(X)**	**63**	43 (51.9%)	11 (−31.0%)	781.9	38.0%	31.8%
(B) with **(Y)**	**63**	42 (54.0%)	11 (−29.7%)	**778.0**	**38.3%**	32.7%
(B) with **(Z)** (chosen combination)	**63**	41 (54.3%)	11 (−23.5%)	791.2	37.3%	**35.4%**
(C) with **(X)**	**63**	**47** (50.9%)	**5** (−105.3%)	966.1	23.4%	30.7%

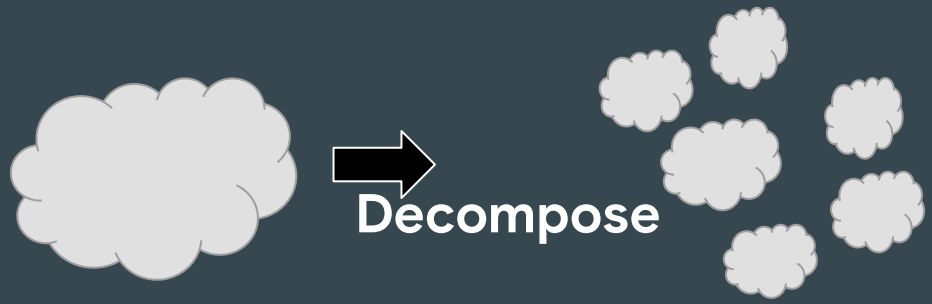
Human-like Programming

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

Sub-problems

```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)
```

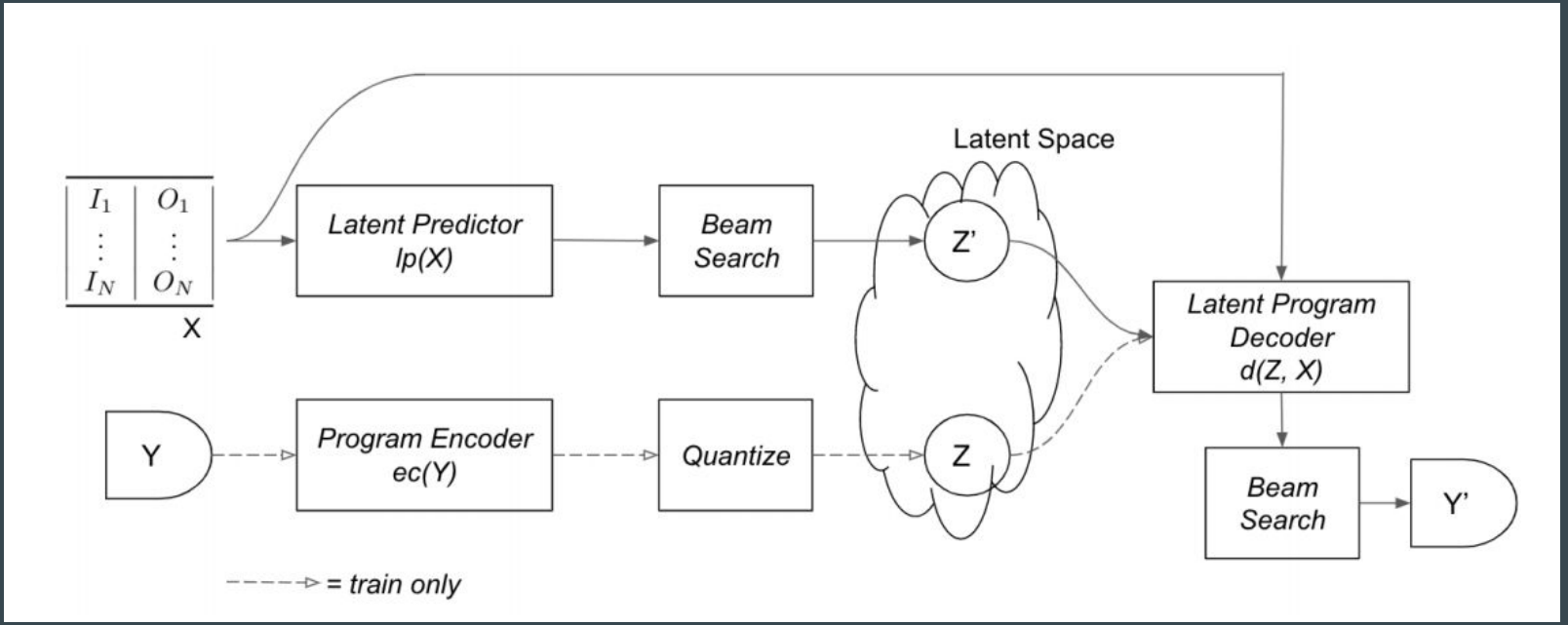
Latent Programmer

Inputs	Outputs
“Jacob,Ethan,James 11"	“11:J.E.J."
“Elijah,Daniel,Aiden 3162"	“3162:E.D.A"
“Rick,Oliver,Mia 26"	“26:R.O.M."
“Mark,Ben,Sam 510"	“510:M.B.S."

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



Two-level Guided Exploration

Length	RobustFill Acc.	LP Acc.
1	**94.5**%	94.0%
2	83.9%	**84.6**%
3	**72.8**%	72.2%
4	63.1%	**66.1**%
5	47.1%	**49.8**%
6	40.6%	**43.0**%
7	30.2%	**34.6**%
8	22.7%	**28.4**%
9	18.6%	**27.0**%
10	14.4%	**25.6**%

Latent Beam Size	Accuracy	Distinct n-Grams			
		n = 1	2	3	4
L = 1	52%	0.13	0.23	0.26	0.28
2	55%	0.13	0.24	0.26	0.28
3	**57**%	0.14	0.25	0.28	0.31
5	57%	0.14	0.26	0.29	0.32
10	56%	**0.14**	**0.26**	**0.30**	**0.33**

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

Input (v)	Output
Mark Henry	M. Henry
Barry M. Myers	B. Myers
Michael Jones	M. Jones
Jon Sanders	J. Sanders

RobustFill

```
Concat(SubStr(0,1), SubStr((Word, 2, Start),
                           (Word, 2, End))
```

Incorrect Decoded Programs

Input (v)	Output	Output’
Mark Henry	M. Henry	M. Henry
Barry M. Myers	B. Myers	B. M
Michael Jones	M. Jones	M. Jones
Jon Sanders	J. Sanders	J. Sanders

RobustFill

```
Concat(SubStr(0,1), SubStr((Word, 2, Start),
                           (Word, 2, End))
```

Beam Search over Program Distribution

Input (v)	Output
Mark Henry	M. Henry
Barry M. Myers	B. Myers
Michael Jones	M. Jones
Jon Sanders	J. Sanders

RobustFill

P_1 P_2 P_3 P_4 . . . P_{10}

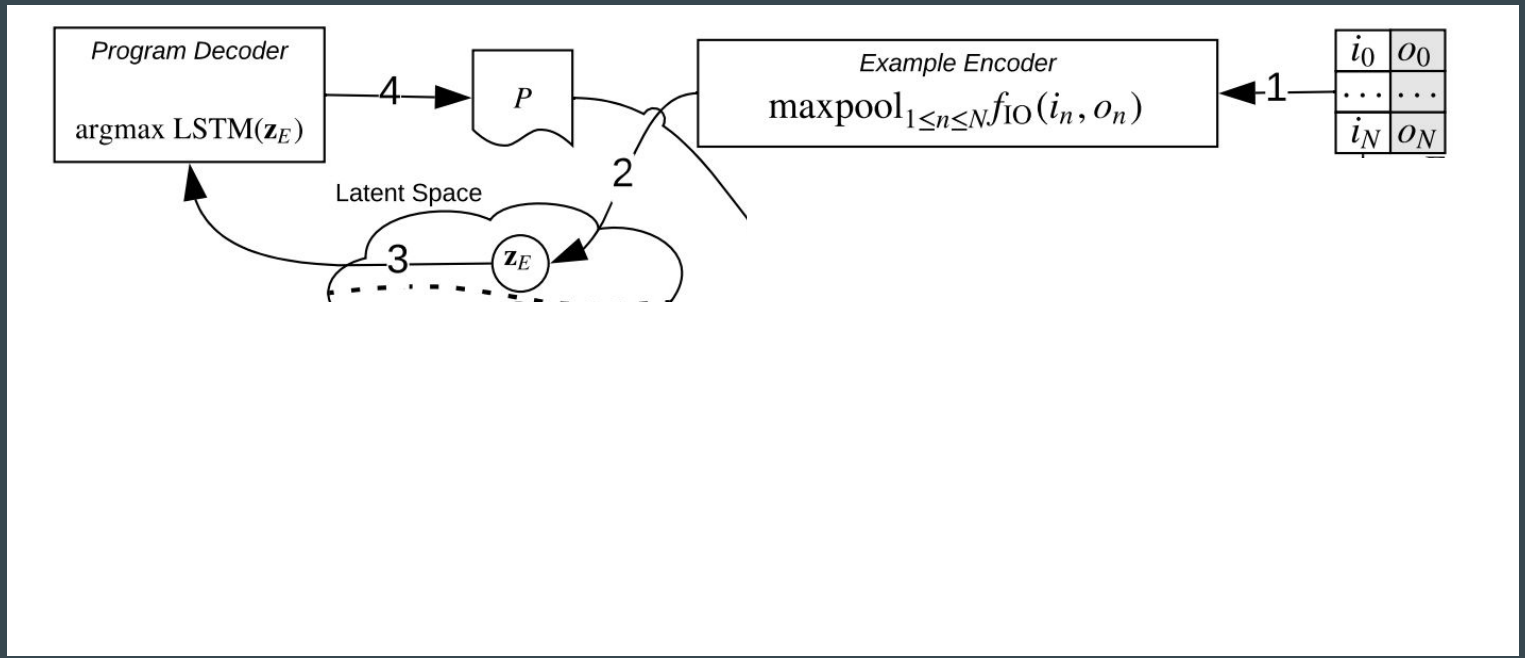
Learning to Fix Mistakes

Input (v)	Output	Output'
Mark Henry	M. Henry	M. Henry
Barry M. Myers	B. Myers	B. M
Michael Jones	M. Jones	M. Jones
Jon Sanders	J. Sanders	J. Sanders

Differentiable Fixer

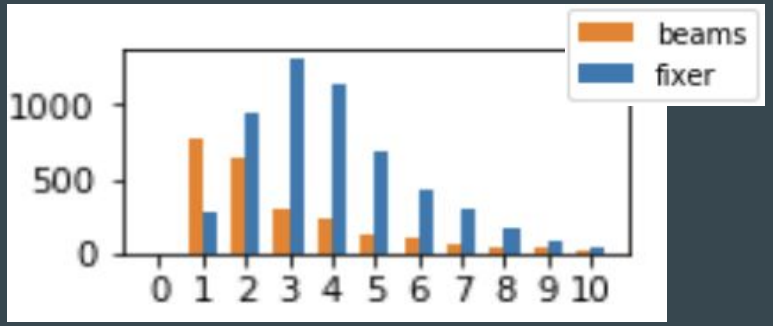
```
Concat(SubStr(0,1), SubStr((Word, -1, Start),
                           (Word, -1, End))
```

Synthesis with Differentiable Fixer

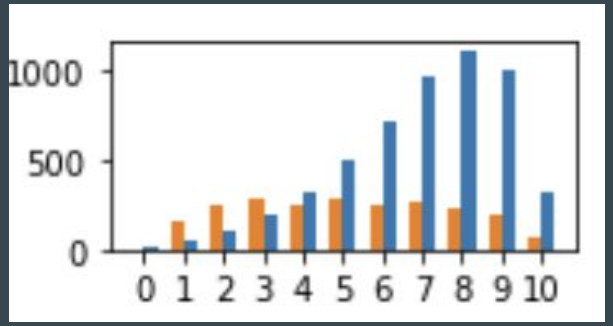


Synthesis with Differentiable Fixer

Algorithm	H	Params	Acc.
Beam	512	$2.2M$	53%
Beam	576	$2.9M$	54%
Beam	608	$3.2M$	56%
Fixer	512	$3.0M$	**61%**



Number of Expressions changed



Fix distance from the End

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