Program Synthesis for Character Level Language Modeling

Pavol Bielik, Veselin Raychev, Martin Vechev

ETH zürich

Character Level Language Model

Statistical language model that estimates a probability distribution over sequences of characters from data

the brown fox jumps over the lazy dog

x_1 ... x_n

Generalized problem statement:

$$\arg\min_{\theta} \frac{1}{n} \sum_{t=1}^{n} \log p(x_t \mid f(x_{<t}, \theta))$$

instantiating function f

our work
Synthesised Program

Neural Network

Probabilistic Grammar

Log-bilinear Model

Our Work

Language Model		Program Synthesis
Probabilistic results		*Deterministic results*
Millions of examples	+	*Hundreds of examples*
Simple, but unexplainable functions		*Interesting programs*

Program Synthesis

Automatically constructs a program that satisfies a given specification (e.g., input/output examples)

Examples

Input		Output
10011011	/	10011000
11100101	/	11100100
00110110	/	00110110

learn function p →

Subject to

```
p(10011011) = 10011000
p(11100101) = 11100100
p(00110110) = 00110110
```

Turn-off the rightmost continuous string of 1 bits

```
p(x) ::=   o1 = (x - 1)
           o2 = (x | o1)
           o3 = (o2 + 1)
           return (o3 & x)
```

Using Programs to Explain Data

Input (sequence of atoms)

' Mg12 He2 Ai13 Fe26 Mg12 Ag47 ...'

Program explaining the input

Probability distribution over next char

[A-Z] [a-z] [0-9] ' '

```
// first character in a word?          model
if Left WriteChar == ' ':             → f1

// how long is current word?
elif PrevChar(' ') WriteDist == 4:    → f2

// second character in a word?
elif Left WriteChar == [A-Z]:         → f3

else:                                 → f4
```

Selects specialized model trained for a particular kind of predictions

Can be further refined to better model the target distribution

What is the model f?

In general:
Any model (e.g., Neural Network)

Our work:
Another program
(LMProgram)

Obtaining Probabilistic Model from a Program

Two step process to obtain a probabilistic model from a learned program g

1. Execute program g(t, $x_{<t}$) to obtain f

t	$x_{<t}$	x_t	g(t, $x_{<t}$)
2	' '	'M'	f4
3	' M'	'g'	f3
4	' Mg'	'1'	f4
5	' Mg1'	'2'	f4
6	' Mg12'	' '	f2
7	' Mg12 '	'H'	f1

2. Calculate the probability $P_f(x_t \mid f(t, x_{<t}))$ using f

P_f estimated using samples assigned to same function f for example, if

```
f4 ::= Left, WriteChar, Left, WriteChar
```

Then $P_{f4}(x_t \mid f4(t, x_{<t})) = P_{f4}(x_t \mid x_{t-1}, x_{t-2})$ corresponds to the tri-gram model obtained using a maximum likelihood estimation (counting)

Program execution

```
f3 ::= Left, WriteChar, PrevPos, Right, WriteChar
```

First execute program g:

g(t, $x_{<t}$) = f3

then execute f3

'Mg12 He2 Ai13 Fe26 M_'

$f3(t, x_{<t})$ = {'M','g'}

used to calculate the probability $P_{f3}(x_t \mid f3(t, x_{<t}))$

DSL for Character Level Language Modeling

Expresses non-trivial data dependencies using instructions that operate on a sequence of characters

Allows conditioning on previously seen input

Allows conditioning on program state

```
TChar ::= SwitchProgram | StateProgram | return LMProgram

LMProgram ::= SimpleProgram
            | SimpleProgram backoff d; LMProgram
            | (SimpleProgram, SimpleProgram)
```

Different model used if current model is not confident in a prediction

Predict a value taken from a previous position in the input

Basic instructions that operate on a sequence of characters

```
SimpleProgram ::=  ε | Move; SimpleProgram | Write; SimpleProgram
Move  ::=  Left, Right, PrevChar(c ∈ Vocabulary), PrevPos
Write ::=  WriteChar, WriteHash, WriteDist
```

Learning a Program

Search technique

SimpleProgram		SwitchProgram
Enumerative search Genetic programming	+ MCMC	ID3+ algorithm
Raychev, V. et. al. *Learning Programs from Noisy Data*. POPL '16, ACM	This work	Raychev, V. et. al. *Probabilistic Model for Code with Decision Trees*. OOPSLA '16, ACM

Problem statement:

$$\arg\min_{g \in \text{TChar}} \frac{1}{n} \sum_{t=1}^{n} \log p(x_t \mid f(t, x_{<t})) + \lambda \cdot \Omega(g)$$

$f = g(t, x_{<t})$

Regularization to avoid too complex programs

Evaluation on Source Code and Natural Language

Datasets

Markus Hutter
http://prize.hutter1.net/
Hutter Prize Wikipedia (Natural Language + Metadata)

characters	*vocabulary size*
100 000 000	*205*

Karpathy, A. et. al.
Visualizing and understanding recurrent networks.
ICLR 2016 Workshop
Linux Kernel (Source Code + Comments)

characters	*vocabulary size*
6 206 996	*101*

Hutter Prize Wikipedia Dataset

Metric	n-gram	This work DSL model	Graves et.al. 2013 Stacked LSTM	Sutskever et.al. 2011 MRNN	We et.al. 2016 MI-LSTM	Chung et.al. 2017 HM-LSTM
BPC	1.94	1.67	1.62	1.60	1.44	1.34

Linux Kernel Dataset

Model	Bits per Character	Error Rate	Training Time	Queries per Second	Model Size
LSTM (2x512)	2.05	38.1%	~80 Hrs	300	53 MB
n-gram (7-gram)	2.23	35.9%	4 Sec	41 000	24 MB
$\text{TChar}_{\text{w/o cache \& backoff}}$	1.92	33.3%	~ 8 Hrs	62 000	17 MB
$\text{TChar}_{\text{w/o backoff}}$	1.84	31.4%	~ 8 Hrs	28 000	19 MB
$\text{TChar}_{\text{w/o cache}}$	1.75	28.0%	~ 8.2 Hrs	24 000	43 MB
TChar	**1.53**	**23.5%**	**~ 8.2 Hrs**	**3 000**	**45 MB**

*Learned specialized programs for **Linux Kernel Dataset***

spaces, first identifier character, semicolons

```
rhp = ACCESS_ONCE(rdp->nocb_head);\n
```

indentation, constants, pointer access, new line

5th International Conference on Learning Representations (ICLR)

Toulon, France, April 24-26 2017