**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.

\[ \text{the brown fox jumps over the lazy dog} \]

Generalized problem statement:

\[ \arg \min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \log p(x_i | f(x_{i-1}, \theta)) \]

**Problem statement:**

Previously seen input

**Genetic programming** selects specialized model trained for

Using programs to explain data

**Using Programs to Explain Data**

' Mg12 He2 Ai13 Fe26 Mg12 Ag47 ...'

Program explaining the input

Probability distribution over next char

**Our Work**

- **Program Synthesis**
  - Automatically constructs a program that satisfies a given specification (e.g., input/output examples)
  - Examples of output program:
    - `Hutter Prize Wikipedia`
    - `Linux Kernel`

**Obtaining Probabilistic Model from a Program**

Two step process to obtain a probabilistic model from a learned program \( g \):

1. Calculate the probability \( P(x_{t-1} | x_t, x_0) \) using \( f \).
2. Calculate the probability \( P(x_t | x_{t-1}) \) using \( f \).

**Evaluation on Source Code and Natural Language**

**Hutter Prize Wikipedia Dataset**

**Metrics**

- **BPC**
  - **1.94**
  - **1.67**
  - **1.62**
  - **1.60**
  - **1.44**
  - **1.34**

**Lincoln Kernel Dataset**

**Metrics**

- **LSTM (2x512)**
  - **2.05**
  - **38.1%**
  - **~80 Hrs**
  - **300**
  - **53 MB**

Learning a Program

**Search technique**

SimpleProgram

Enumerative search

Problem statement:

\[ \arg \min_{g \in \text{TChar}} \frac{1}{n} \sum_{i=1}^{n} \log p(x_i | f(t, x_{i-1})) + \lambda \cdot |g| \]

Regularization to avoid too complex programs

---

**Using Programs to Explain Data**

<table>
<thead>
<tr>
<th>Input (sequence of atoms)</th>
<th>( ' Mg12 He2 Ai13 Fe26 Mg12 Ag47 ...' )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program explaining the input</td>
<td>Probability distribution over next char</td>
</tr>
<tr>
<td>// first character in a word?</td>
<td>TChar := SwitchProgram</td>
</tr>
<tr>
<td>// how long is current word?</td>
<td>TChar := SimpleProgram</td>
</tr>
<tr>
<td>( g(t, x) ) to obtain ( f )</td>
<td>Predict a value taken from a previous position in the input</td>
</tr>
<tr>
<td>if Left WriteChar == ( 'v' )</td>
<td>Different model used if current model is not confident in a prediction</td>
</tr>
<tr>
<td>elseif PrevChar(' ') WriteDist := 4;</td>
<td>LMProgram := SimpleProgram</td>
</tr>
<tr>
<td>// second character in a word?</td>
<td>LMProgram := SimpleProgram</td>
</tr>
<tr>
<td>else:</td>
<td>(SimpleProgram, SimpleProgram)</td>
</tr>
</tbody>
</table>

**Our Work**

- **Program Synthesis**
  - Automatically constructs a program that satisfies a given specification (e.g., input/output examples)
  - Examples of output program:
    - `Hutter Prize Wikipedia`
    - `Linux Kernel`

**Obtaining Probabilistic Model from a Program**

Two step process to obtain a probabilistic model from a learned program \( g \):

1. Calculate the probability \( P(x_{t-1} | x_t, x_0) \) using \( f \).
2. Calculate the probability \( P(x_t | x_{t-1}) \) using \( f \).

\( p_t \) estimated using samples assigned to same function \( f \) for example, \( f \).

**Evaluation on Source Code and Natural Language**

**Hutter Prize Wikipedia Dataset**

<table>
<thead>
<tr>
<th>Metric</th>
<th>n-gram</th>
<th>DSL model</th>
<th>Stackced LSTM</th>
<th>MRNN</th>
<th>Mi-LSTM</th>
<th>HM-LSTM</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPC</td>
<td>1.94</td>
<td>1.67</td>
<td>1.62</td>
<td>1.60</td>
<td>1.44</td>
<td>1.34</td>
</tr>
</tbody>
</table>

**Lincoln Kernel Dataset**

<table>
<thead>
<tr>
<th>Model</th>
<th>Bits per Character</th>
<th>Error Rate</th>
<th>Training Time</th>
<th>Queries per Second</th>
<th>Model Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM 2x512</td>
<td>2.05</td>
<td>38.1%</td>
<td>~80 Hrs</td>
<td>300</td>
<td>53 MB</td>
</tr>
<tr>
<td>n-gram (7-gram)</td>
<td>2.23</td>
<td>35.9%</td>
<td>4 Sec</td>
<td>41 000</td>
<td>24 MB</td>
</tr>
<tr>
<td>TChar (cache &amp; backoff)</td>
<td>1.92</td>
<td>33.3%</td>
<td>~8 Hrs</td>
<td>62 000</td>
<td>17 MB</td>
</tr>
<tr>
<td>TChar (cache)</td>
<td>1.84</td>
<td>31.4%</td>
<td>~8 Hrs</td>
<td>28 000</td>
<td>19 MB</td>
</tr>
<tr>
<td>TChar (cache)</td>
<td>1.75</td>
<td>28.0%</td>
<td>~8 Hrs</td>
<td>24 000</td>
<td>43 MB</td>
</tr>
<tr>
<td>TChar</td>
<td>1.53</td>
<td>23.5%</td>
<td>~8 Hrs</td>
<td>3 000</td>
<td>45 MB</td>
</tr>
</tbody>
</table>

**Learned specialized programs for Linux Kernel Dataset**

```
chp = ACCESS_NONE

int insertion = constants;

pointer access = new time;
```