Exploring the Naturalness of Code with Recurrent Neural Networks

Jack Lanchantin and Ji Gao

April 26, 2016
Background/Motivation

“The cat sat apple the mat.” → High entropy (unnatural)

“The cat sat on the mat.” → Low entropy (natural)

Hypothesis: “unnatural” code is suspicious, possibly suggesting a bug
Language Models

\[ P(S) = P(t_1, t_2, \ldots, t_N) = P(t_1) \cdot \prod_{i=2}^{N} P(t_i | t_1, \ldots, t_{i-1}) \]

Hard to compute
n-gram Language Models

\[ P(S) = P(t_1, t_2, \ldots, t_N) = P(t_1) \cdot \prod_{i=2}^{N} P(t_i | t_1, \ldots, t_{i-1}) \]

\[ P_{\text{ngram}}(t_i | h) = P(t_i | t_{i-n+1}, \ldots, t_{i-1}) \]
Entropy (information theory)

Entropy (H) measures the amount of uncertainty in a distribution

\[ H = - \sum p(x) \log p(x) \]

English text has between 0.6 and 1.3 bits of entropy for each character

**Goal:** Find the entropy of a new line from a previously trained language model in order to determine if it is a possibly buggy line
Goal: Test more complex language models and see if we can better predict buggy lines based on entropy

Method: We use Recurrent Neural Networks as our language model which can more accurately handle long term dependencies in the language
Recurrent Neural Networks (RNNs)

Traditional Neural Network

- Input
- Hidden
- Output

Recurrent Neural Network

- Input
- Hidden
- Output
  - Recurrent connection (arrow)
Recurrent Neural Networks (RNNs)

Recurrent nets can model the **full** conditional distribution

\[ P(S) = P(t_1, t_2, \ldots, t_N) = P(t_1) \cdot \prod_{i=2}^{N} P(t_i | t_1, \ldots, t_{i-1}) \]

... at the cost of a much higher optimization problem
Character-level RNN Language Model
Character-level RNN Language Model

\[
\hat{Y} \rightarrow \\
\begin{array}{cccc}
1.0 & 0.5 & 0.1 & -1.0 \\
2.2 & 0.3 & 0.5 & 1.1 \\
-3.0 & -1.0 & 1.9 & 0.2 \\
4.1 & 1.2 & -1.1 & 2.2 \\
\end{array}
\]

\[
\begin{array}{cccc}
1.0 & 0.5 & 0.1 & -1.0 \\
2.2 & 0.3 & 0.5 & 1.1 \\
-3.0 & -1.0 & 1.9 & 0.2 \\
4.1 & 1.2 & -1.1 & 2.2 \\
\end{array}
\]

\[
\begin{array}{cccc}
1.0 & 0.5 & 0.1 & -1.0 \\
2.2 & 0.3 & 0.5 & 1.1 \\
-3.0 & -1.0 & 1.9 & 0.2 \\
4.1 & 1.2 & -1.1 & 2.2 \\
\end{array}
\]

\[
\begin{array}{cccc}
1.0 & 0.5 & 0.1 & -1.0 \\
2.2 & 0.3 & 0.5 & 1.1 \\
-3.0 & -1.0 & 1.9 & 0.2 \\
4.1 & 1.2 & -1.1 & 2.2 \\
\end{array}
\]

\[
\begin{array}{cccc}
1.0 & 0.5 & 0.1 & -1.0 \\
2.2 & 0.3 & 0.5 & 1.1 \\
-3.0 & -1.0 & 1.9 & 0.2 \\
4.1 & 1.2 & -1.1 & 2.2 \\
\end{array}
\]

\[
\begin{array}{cccc}
1.0 & 0.5 & 0.1 & -1.0 \\
2.2 & 0.3 & 0.5 & 1.1 \\
-3.0 & -1.0 & 1.9 & 0.2 \\
4.1 & 1.2 & -1.1 & 2.2 \\
\end{array}
\]
$\hat{Y}$

\[
\begin{array}{c}
1.0 \\
2.2 \\
-3.0 \\
4.1 \\
\end{array}
\begin{array}{c}
0 \\
1 \\
0 \\
0 \\
\end{array}
\]

\[
\begin{array}{c}
0.5 \\
0.3 \\
-1.0 \\
1.2 \\
\end{array}
\begin{array}{c}
0 \\
0 \\
1 \\
0 \\
\end{array}
\]

\[
\begin{array}{c}
0.1 \\
0.5 \\
1.9 \\
-1.1 \\
\end{array}
\begin{array}{c}
0 \\
0 \\
1 \\
0 \\
\end{array}
\]

\[
\begin{array}{c}
-1.0 \\
1.1 \\
0.2 \\
2.2 \\
\end{array}
\begin{array}{c}
0 \\
0 \\
0 \\
1 \\
\end{array}
\]

X

\[
\begin{array}{c}
1 \\
0 \\
0 \\
0 \\
\end{array}
\begin{array}{c}
0 \\
1 \\
0 \\
0 \\
\end{array}
\begin{array}{c}
0 \\
0 \\
1 \\
0 \\
\end{array}
\begin{array}{c}
0 \\
0 \\
1 \\
0 \\
\end{array}
\]

$W_x$

$W_y$

$W_h$

$Ŷ$

$X$

$h$

$e$

$l$

$l$
\[ P(s_2 | s_1) \quad P(s_3 | s_1, s_2) \quad P(s_4 | s_1, s_2, s_3) \quad P(s_5 | s_1, \ldots, s_4) \]

\[ \begin{align*}
  P(s_2 | s_1) & \\
  P(s_3 | s_1, s_2) & \\
  P(s_4 | s_1, s_2, s_3) & \\
  P(s_5 | s_1, \ldots, s_4) & 
\end{align*} \]
\[
\hat{Y} \rightarrow
\]

\[
\begin{array}{c}
\text{loss}(t = 1) \\
\begin{array}{cccc}
1.0 & 0 & 0 & 0 \\
2.2 & 0 & 0 & 0 \\
-3.0 & 0 & 0 & 0 \\
4.1 & 0 & 0 & 0 \\
\end{array}
\end{array}
\]

\[
\begin{array}{c}
\text{loss}(t = 2) \\
\begin{array}{cccc}
0.5 & 0.5 & 0.3 & 0.1 \\
0.3 & 0.3 & 1.9 & 0.5 \\
-1.0 & -1.0 & 1.9 & 0.5 \\
1.2 & 1.2 & -1.1 & 1.9 \\
\end{array}
\end{array}
\]

\[
\begin{array}{c}
\text{loss}(t = 3) \\
\begin{array}{cccc}
0.1 & 0.1 & 0.5 & -1.0 \\
0.5 & 0.5 & 0.5 & 1.1 \\
0.5 & 0.5 & 0.5 & 1.1 \\
-1.1 & -1.1 & -1.1 & -1.1 \\
\end{array}
\end{array}
\]

\[
\begin{array}{c}
\text{loss}(t = 4) \\
\begin{array}{cccc}
-1.0 & -1.0 & 0.2 & 2.2 \\
1.1 & 1.1 & 0.2 & 2.2 \\
0.2 & 0.2 & 0.2 & 0.2 \\
2.2 & 2.2 & 2.2 & 2.2 \\
\end{array}
\end{array}
\]

\[
X \rightarrow
\]

\[
\begin{array}{c}
\text{h} \\
\begin{array}{cc}
1 & 0 \\
0 & 0 \\
\end{array}
\end{array}
\]

\[
\begin{array}{c}
e \\
\begin{array}{cc}
0 & 0 \\
0 & 0 \\
\end{array}
\end{array}
\]

\[
\begin{array}{c}
l \\
\begin{array}{cc}
0 & 0 \\
0 & 0 \\
\end{array}
\end{array}
\]

\[
\begin{array}{c}
l \\
\begin{array}{cc}
0 & 0 \\
0 & 0 \\
\end{array}
\end{array}
\]

\[
W_x \quad W_y \quad W_h \quad W_x \quad W_y \quad W_h \quad W_x \quad W_y
\]
Long Short Term Memory Networks (LSTMs)

Recurrent networks suffer from the “vanishing gradient problem”

- Aren’t able to model long term dependencies in sequences
Long Short Term Memory Networks (LSTMs)

Recurrent networks suffer from the “vanishing gradient problem”
- Aren’t able to model long term dependencies in sequences

Use “gating units” to learn when to remember
We train two models:

- Global language model (GLM)
- Local language model (LLM)

To get the final entropy, we evaluate

- the entropy from the global model
- the combined entropy of the local and global model:

\[ H_{total} = \lambda H_{GLM} + (1-\lambda)H_{LLM} \]
## Dataset

Elasticsearch project on Github

<table>
<thead>
<tr>
<th></th>
<th>Snapshots</th>
<th>JAVA files</th>
<th>Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training set</td>
<td>50</td>
<td>118,164</td>
<td>16,502,732</td>
</tr>
<tr>
<td>Testing set*</td>
<td>18</td>
<td>59,180</td>
<td>9,437,902</td>
</tr>
<tr>
<td>Total</td>
<td>68</td>
<td>177,344</td>
<td>25,940,634</td>
</tr>
</tbody>
</table>

*To actually test, we selected 10,902 buggy lines and 10,902 non-buggy lines from the total 9,437,902 lines

The “local LM” was trained on the remaining 9,416,098 lines
Dataset

- Global LM
- Local LM
- Test lines
## Average Entropy

<table>
<thead>
<tr>
<th></th>
<th>Buggy Lines</th>
<th>Non-Buggy Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Global LSTM</td>
<td>Global LSTM</td>
</tr>
<tr>
<td>Avg Entropy</td>
<td>1.6498</td>
<td>1.5925</td>
</tr>
<tr>
<td></td>
<td>Global + Local LSTM</td>
<td>Global + Local LSTM</td>
</tr>
<tr>
<td>Avg Entropy</td>
<td>1.5842</td>
<td>1.5230</td>
</tr>
</tbody>
</table>

**Avg Entropy Difference (buggy avg - nonbuggy avg)**

- **Global LSTM**: 0.064
- **Global + Local LSTM**: 0.067
AUC comparison

- n-gram
- Global LSTM
- Global + Local LSTM
Trained a “buggy language model” on the buggy lines (± 5 lines) on the training set.

Tested on the same lines as before

<table>
<thead>
<tr>
<th></th>
<th>Buggy Lines</th>
<th>Non-Buggy Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg Entropy</td>
<td>1.20028</td>
<td>1.18082</td>
</tr>
</tbody>
</table>
Future Work

1. Train the language model on many different projects of the same language (e.g. Java) in order to create a true model of the actual language.

2. Then, train a local language model on just the project of interest.