Building Language Models for Text with Named Entities
Md Rizwan Parvez¹, Saikat Chakraborty², Baishakhi Ray², Kai-Wei Chang¹
{rizwan, kwchang}@cs.ucla.edu¹, {sc2nf, br8jr}@virginia.edu²

Motivation

• Generating text or code with a large number of named entities (e.g., different variable names in source code) is a very challenging problem for any language model due to their wide variations.

• Existing language models:
  - all to address this problem
  - focus on copying/matching the entity names from the reference corpus
  - Therefore, can not be used for general purpose (free form) language generation applications like text auto-completion, spelling correction, code suggestion, etc.

• To address this problem, we propose a novel language model for texts with many entity names by leveraging the deterministic entity type of the named entities.

Methodology

Our model learns learns the probability distribution over the candidate words by decomposing it into two sub-problems:

• **Type Model (θ_t):**
  - considers all of them equal (e.g., olive oil, canola oil, grape oil --all are different varieties of oil) and represent them by their type information.
  - reduces the vocab size to a great extent
  - predicts the type s(w) of each entity (w) more accurately.

• **Entity Composite Model (θ_e):**
  - uses the entity type generated by the type model as a prior
  - calculates the conditional probability distribution of the actual entity names.

We depict these two models in Figure 1.

![Language Model (entity composite type model)](image)

Figure 1. An illustration of our model. Given a context, the **type model** (in bottom red block) generates the type of the words. Further, for that given context & type of each candidate generated by the **type model**, the **entity composite model** (in upper green block) generates the conditional actual entity names.

$$P(w / \bar{w}, s(w), \theta_e) = P(s(w) / \bar{w}, \theta_t) \times \left\{ P(s(w) / \bar{w}, \theta_t) \right\}$$

where S is set of all entity types.

Experimental Results

To evaluate our model, we create two benchmark datasets that involve many named entities.

• Dataset-1:
  - 95,786 cooking recipes
  - Manually categorized 8 super-ingredients (i.e., types); e.g., proteins, vegetables, fruits.
  - The generation performance:

<table>
<thead>
<tr>
<th>Model</th>
<th>corpus</th>
<th>Vocab</th>
<th>Perplexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWD_LSTM</td>
<td>original</td>
<td>52,742</td>
<td>20.23</td>
</tr>
<tr>
<td>AWD_LSTM</td>
<td>type</td>
<td>51,675</td>
<td>17.62</td>
</tr>
<tr>
<td>AWD_LSTM</td>
<td>original</td>
<td>52,742</td>
<td>20.26</td>
</tr>
<tr>
<td>Our model</td>
<td>original</td>
<td>52,742</td>
<td>9.67</td>
</tr>
</tbody>
</table>

• Accuracy improvement by our model:

<table>
<thead>
<tr>
<th>Entity name</th>
<th>Training frequency</th>
<th>#Blanks</th>
<th>Accuracy</th>
<th>MCQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Milk</td>
<td>14,136</td>
<td>4001</td>
<td>59.34</td>
<td>94.90</td>
</tr>
<tr>
<td>Salt</td>
<td>33,306</td>
<td>9,888</td>
<td>62.47</td>
<td>89.29</td>
</tr>
<tr>
<td>Apple</td>
<td>7,205</td>
<td>720</td>
<td>30.28</td>
<td>89.86</td>
</tr>
<tr>
<td>Breads</td>
<td>11,673</td>
<td>3,074</td>
<td>52.64</td>
<td>94.53</td>
</tr>
<tr>
<td>Tomato</td>
<td>19,875</td>
<td>6,224</td>
<td>45.24</td>
<td>77.70</td>
</tr>
</tbody>
</table>

• Dataset-2:
  - 500 open source Android projects collected from GitHub
  - Abstract Syntax Tree (AST) based entity type
  - 22.06% better in perplexity with respect to state-of-art code generation baseline SLP-Core³.

References