Spell Once, Summon Anywhere:  
A Two-Level Open-Vocabulary Language Model  
AAAI 2019 Technical Track

Sebastian J. Mielke and Jason Eisner

sjmielke@jhu.edu, jason@cs.jhu.edu  
Department of Computer Science, Johns Hopkins University, Baltimore, MD, USA
Language modeling: a generative story of text

\[ p(\text{the cat chased the}) \]
Language modeling: a generative story of text

\[ p(\text{the cat chased the}) = p(\text{the}) \]
Language modeling: a generative story of text

\[
p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} | \text{the})
\]
Language modeling: a generative story of text

\[ p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} | \text{the}) \cdot p(\text{chased} | \text{the cat}) \]
Language modeling: a generative story of text

\[ p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} \mid \text{the}) \cdot p(\text{chased} \mid \text{the cat}) \cdot p(\text{the} \mid \text{the cat chased}) \]
Language modeling: a generative story of text

\[ p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} | \text{the}) \cdot p(\text{chased} | \text{the cat}) \cdot p(\text{the} | \text{the cat chased}) \]

**Lexicon / vocabulary**

<table>
<thead>
<tr>
<th>type</th>
<th>spelling</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w )</td>
<td>( \sigma(w) )</td>
</tr>
<tr>
<td>1</td>
<td>the</td>
</tr>
<tr>
<td>2</td>
<td>cat</td>
</tr>
<tr>
<td>3</td>
<td>chased</td>
</tr>
</tbody>
</table>
Language modeling: a generative story of text

\[ p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} \mid \text{the}) \cdot p(\text{chased} \mid \text{the cat}) \cdot p(\text{the} \mid \text{the cat chased}) \]

**Lexicon / vocabulary**

<table>
<thead>
<tr>
<th>type</th>
<th>spelling</th>
<th>embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>[w]</td>
<td>[\sigma(w)]</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>the</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>cat</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>chased</td>
<td></td>
</tr>
</tbody>
</table>
Language modeling: a generative story of text

\[ p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat | the}) \cdot p(\text{chased | the cat}) \cdot p(\text{the | the cat chased}) \]

**Lexicon / vocabulary**

<table>
<thead>
<tr>
<th>type</th>
<th>spelling</th>
<th>embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>the</td>
<td>[0.2, ⋯, 0.0]</td>
</tr>
<tr>
<td>2</td>
<td>cat</td>
<td>[0.4, ⋯, 0.5]</td>
</tr>
<tr>
<td>3</td>
<td>chased</td>
<td>[−0.1, ⋯, 0.2]</td>
</tr>
</tbody>
</table>
Language modeling: a generative story of text

\[
p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} | \text{the}) \cdot p(\text{chased} | \text{the cat}) \cdot p(\text{the} | \text{the cat chased})
\]

### Lexicon / vocabulary

<table>
<thead>
<tr>
<th>type</th>
<th>spelling</th>
<th>embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>(w)</td>
<td>(\sigma(w))</td>
<td>(e(w))</td>
</tr>
<tr>
<td>1</td>
<td>the</td>
<td>([0.2, \cdots, 0.0])</td>
</tr>
<tr>
<td>2</td>
<td>cat</td>
<td>([0.4, \cdots, 0.5])</td>
</tr>
<tr>
<td>3</td>
<td>chased</td>
<td>([-0.1, \cdots, 0.2])</td>
</tr>
</tbody>
</table>

### Text generation with an RNN

...but what is the word?

Pure character-level model as the solution?
Language modeling: a generative story of text

\[ p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} \mid \text{the}) \cdot p(\text{chased} \mid \text{the cat}) \cdot p(\text{the} \mid \text{the cat chased}) \]

<table>
<thead>
<tr>
<th>type</th>
<th>spelling</th>
<th>embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>the</td>
<td>[0.2, ⋯, 0.0]</td>
</tr>
<tr>
<td>2</td>
<td>cat</td>
<td>[0.4, ⋯, 0.5]</td>
</tr>
<tr>
<td>3</td>
<td>chased</td>
<td>[−0.1, ⋯, 0.2]</td>
</tr>
</tbody>
</table>

Text generation with an RNN

...but what is the word?

Pure character-level model as the solution?
Language modeling: a generative story of text

\[ p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} | \text{the}) \cdot p(\text{chased} | \text{the cat}) \cdot p(\text{the} | \text{the cat chased}) \]

**Lexicon / vocabulary**

<table>
<thead>
<tr>
<th>type</th>
<th>spelling</th>
<th>embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>(w)</td>
<td>(\sigma(w))</td>
<td>(e(w))</td>
</tr>
<tr>
<td>1</td>
<td>the</td>
<td>[0.2, \ldots, 0.0]</td>
</tr>
<tr>
<td>2</td>
<td>cat</td>
<td>[0.4, \ldots, 0.5]</td>
</tr>
<tr>
<td>3</td>
<td>chased</td>
<td>[−0.1, \ldots, 0.2]</td>
</tr>
</tbody>
</table>

**Text generation with an RNN**

...but what is the word?

Pure character-level model as the solution?
Language modeling: a generative story of text

\[ p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} | \text{the}) \cdot p(\text{chased} | \text{the cat}) \cdot p(\text{the} | \text{the cat chased}) \]

**Lexicon / vocabulary**

<table>
<thead>
<tr>
<th>type ( w )</th>
<th>spelling ( \sigma(w) )</th>
<th>embedding ( e(w) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>the</td>
<td>[0.2, ⋯, 0.0]</td>
</tr>
<tr>
<td>2</td>
<td>cat</td>
<td>[0.4, ⋯, 0.5]</td>
</tr>
<tr>
<td>3</td>
<td>chased</td>
<td>[−0.1, ⋯, 0.2]</td>
</tr>
</tbody>
</table>

**Text generation with an RNN**

...but what is the word?

Pure character-level model as the solution?
Language modeling: a generative story of text

\[ p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat | the}) \cdot p(\text{chased | the cat}) \cdot p(\text{the | the cat chased}) \]

**Lexicon / vocabulary**

<table>
<thead>
<tr>
<th>type</th>
<th>spelling</th>
<th>embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>the</td>
<td>[0.2, ..., 0.0]</td>
</tr>
<tr>
<td>2</td>
<td>cat</td>
<td>[0.4, ..., 0.5]</td>
</tr>
<tr>
<td>3</td>
<td>chased</td>
<td>[−0.1, ..., 0.2]</td>
</tr>
</tbody>
</table>

**Text generation with an RNN**

The process of text generation involves feeding each word through the RNN cells, starting with the initial state and updating the hidden state at each step. The final state is then used to predict the next word in the sequence.
Language modeling: a generative story of text

\[ p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} | \text{the}) \cdot p(\text{chased} | \text{the cat}) \cdot p(\text{the} | \text{the cat chased}) \]

**Lexicon / vocabulary**

<table>
<thead>
<tr>
<th>type</th>
<th>spelling</th>
<th>embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w )</td>
<td>( \sigma(w) )</td>
<td>( e(w) )</td>
</tr>
<tr>
<td>1</td>
<td>the</td>
<td>([0.2, \ldots, 0.0])</td>
</tr>
<tr>
<td>2</td>
<td>cat</td>
<td>([0.4, \ldots, 0.5])</td>
</tr>
<tr>
<td>3</td>
<td>chased</td>
<td>([-0.1, \ldots, 0.2])</td>
</tr>
</tbody>
</table>

**Text generation with an RNN**
Language modeling: a generative story of text

\[ p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat } | \text{ the}) \cdot p(\text{chased } | \text{ the cat}) \cdot p(\text{the } | \text{ the cat chased}) \]

**Lexicon / vocabulary**

<table>
<thead>
<tr>
<th>type</th>
<th>spelling</th>
<th>embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>the</td>
<td>[0.2, ⋮, 0.0]</td>
</tr>
<tr>
<td>2</td>
<td>cat</td>
<td>[0.4, ⋮, 0.5]</td>
</tr>
<tr>
<td>3</td>
<td>chased</td>
<td>[−0.1, ⋮, 0.2]</td>
</tr>
</tbody>
</table>

**Text generation with an RNN**

The text generation process involves an RNN cell for each word in the sentence. The sequence is 

\[ \text{the} \xrightarrow{\text{RNN cell}} \text{cat} \xrightarrow{\text{RNN cell}} \text{chased} \]
Language modeling: a generative story of text

\[ p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} | \text{the}) \cdot p(\text{chased} | \text{the cat}) \cdot p(\text{the} | \text{the cat chased}) \]

### Lexicon / vocabulary

<table>
<thead>
<tr>
<th>type</th>
<th>spelling</th>
<th>embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w )</td>
<td>( \sigma(w) )</td>
<td>( e(w) )</td>
</tr>
<tr>
<td>1</td>
<td>the</td>
<td>([0.2, \cdots, 0.0])</td>
</tr>
<tr>
<td>2</td>
<td>cat</td>
<td>([0.4, \cdots, 0.5])</td>
</tr>
<tr>
<td>3</td>
<td>chased</td>
<td>([-0.1, \cdots, 0.2])</td>
</tr>
</tbody>
</table>

### Text generation with an RNN

- **RNN cell**
- **the**
- **cat**
- **chased**
- **the**

...but what is the word?

Pure character-level model as the solution?

...can't we memorize it?
Language modeling: a generative story of text

\[ p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} | \text{the}) \cdot p(\text{chased} | \text{the cat}) \cdot p(\text{the} | \text{the cat chased}) \]

**Lexicon / vocabulary**

<table>
<thead>
<tr>
<th>type</th>
<th>spelling</th>
<th>embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>the</td>
<td>[0.2, \cdots, 0.0]]</td>
</tr>
<tr>
<td>2</td>
<td>cat</td>
<td>[0.4, \cdots, 0.5]]</td>
</tr>
<tr>
<td>3</td>
<td>chased</td>
<td>[-0.1, \cdots, 0.2]</td>
</tr>
</tbody>
</table>

**Text generation with an RNN**

...but what is the word? Pure character-level model as the solution...
Language modeling: a generative story of text

\[ p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat }|\text{ the}) \cdot p(\text{chased }|\text{ the cat}) \cdot p(\text{the }|\text{ the cat chased}) \]

**Lexicon / vocabulary**

<table>
<thead>
<tr>
<th>Type</th>
<th>Spelling</th>
<th>Embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>the</td>
<td>[0.2, ⋯, 0.0]</td>
</tr>
<tr>
<td>2</td>
<td>cat</td>
<td>[0.4, ⋯, 0.5]</td>
</tr>
<tr>
<td>3</td>
<td>chased</td>
<td>[−0.1, ⋯, 0.2]</td>
</tr>
<tr>
<td>4</td>
<td>UNK</td>
<td>[0.3, ⋯, 0.1]</td>
</tr>
</tbody>
</table>

**Text generation with an RNN**
Language modeling: a generative story of text

\[ p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat}) \cdot p(\text{chased}) \cdot p(\text{the} \mid \text{the cat chased}) \]

**Lexicon / vocabulary**

<table>
<thead>
<tr>
<th>type</th>
<th>spelling</th>
<th>embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>the</td>
<td>[0.2, ⋯, 0.0]</td>
</tr>
<tr>
<td>2</td>
<td>cat</td>
<td>[0.4, ⋯, 0.5]</td>
</tr>
<tr>
<td>3</td>
<td>chased</td>
<td>[−0.1, ⋯, 0.2]</td>
</tr>
<tr>
<td>4</td>
<td>UNK</td>
<td>[0.3, ⋯, 0.1]</td>
</tr>
</tbody>
</table>

**Text generation with an RNN**

...but what is the word?
Language modeling: a generative story of text

\[ p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} | \text{the}) \cdot p(\text{chased} | \text{the cat}) \cdot p(\text{the} | \text{the cat chased}) \]

**Lexicon / vocabulary**

<table>
<thead>
<tr>
<th>type</th>
<th>spelling</th>
<th>embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>\text{the}</td>
<td>[0.2, \ldots, 0.0]</td>
</tr>
<tr>
<td>2</td>
<td>\text{cat}</td>
<td>[0.4, \ldots, 0.5]</td>
</tr>
<tr>
<td>3</td>
<td>\text{chased}</td>
<td>[-0.1, \ldots, 0.2]</td>
</tr>
<tr>
<td>4</td>
<td>\text{UNK}</td>
<td>[0.3, \ldots, 0.1]</td>
</tr>
</tbody>
</table>

**Text generation with an RNN**

...but what is the word?

**Pure character-level model as the solution?**
Language modeling: a generative story of text

\[ p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} | \text{the}) \cdot p(\text{chased} | \text{the cat}) \cdot p(\text{the} | \text{the cat chased}) \]

**Lexicon / vocabulary**

<table>
<thead>
<tr>
<th>type</th>
<th>spelling</th>
<th>embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>the</td>
<td>[0.2, ⋯, 0.0]</td>
</tr>
<tr>
<td>2</td>
<td>cat</td>
<td>[0.4, ⋯, 0.5]</td>
</tr>
<tr>
<td>3</td>
<td>chased</td>
<td>[−0.1, ⋯, 0.2]</td>
</tr>
<tr>
<td>4</td>
<td>UNK</td>
<td>[0.3, ⋯, 0.1]</td>
</tr>
</tbody>
</table>

**Text generation with an RNN**

...but what is the word?

**Pure character-level model as the solution?**
Language modeling: a generative story of text

\[
p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} | \text{the}) \cdot p(\text{chased} | \text{the cat}) \cdot p(\text{the} | \text{the cat chased})
\]

**Lexicon / vocabulary**

<table>
<thead>
<tr>
<th>type</th>
<th>spelling</th>
<th>embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ( w )</td>
<td>the ( \sigma(w) )</td>
<td>[0.2, ⋯, 0.0]</td>
</tr>
<tr>
<td>2</td>
<td>cat ( \sigma(w) )</td>
<td>[0.4, ⋯, 0.5]</td>
</tr>
<tr>
<td>3</td>
<td>chased ( \sigma(w) )</td>
<td>[−0.1, ⋯, 0.2]</td>
</tr>
<tr>
<td>4</td>
<td>UNK ( \sigma(w) )</td>
<td>[0.3, ⋯, 0.1]</td>
</tr>
</tbody>
</table>

**Text generation with an RNN**

...but what is the word?

**Pure character-level model as the solution?**

\[\text{the} \quad \text{cat} \quad \text{chased} \quad \text{th} \]
Language modeling: a generative story of text

\[ p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} \mid \text{the}) \cdot p(\text{chased} \mid \text{the cat}) \cdot p(\text{the} \mid \text{the cat chased}) \]

**Lexicon / vocabulary**

<table>
<thead>
<tr>
<th>type</th>
<th>spelling</th>
<th>embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>(w)</td>
<td>(\sigma(w))</td>
<td>(e(w))</td>
</tr>
<tr>
<td>1</td>
<td>the</td>
<td>[0.2, ..., 0.0]</td>
</tr>
<tr>
<td>2</td>
<td>cat</td>
<td>[0.4, ..., 0.5]</td>
</tr>
<tr>
<td>3</td>
<td>chased</td>
<td>[−0.1, ..., 0.2]</td>
</tr>
<tr>
<td>4</td>
<td>UNK</td>
<td>[0.3, ..., 0.1]</td>
</tr>
</tbody>
</table>

**Text generation with an RNN**

...but what is the word?

**Pure character-level model as the solution?**
Language modeling: a generative story of text

\[ p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} \mid \text{the}) \cdot p(\text{chased} \mid \text{the cat}) \cdot p(\text{the} \mid \text{the cat chased}) \]

**Lexicon / vocabulary**

<table>
<thead>
<tr>
<th>type</th>
<th>spelling</th>
<th>embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>the</td>
<td>[0.2, \cdots, 0.0]</td>
</tr>
<tr>
<td>2</td>
<td>cat</td>
<td>[0.4, \cdots, 0.5]</td>
</tr>
<tr>
<td>3</td>
<td>chased</td>
<td>[−0.1, \cdots, 0.2]</td>
</tr>
<tr>
<td>4</td>
<td>UNK</td>
<td>[0.3, \cdots, 0.1]</td>
</tr>
</tbody>
</table>

**Text generation with an RNN**

...but what is the word?

Pure character-level model as the solution?

Ugh, spelling the again...
Language modeling: a generative story of text

\[ p(\text{the cat chased the}) = p(\text{the}) \cdot p(\text{cat} | \text{the}) \cdot p(\text{chased} | \text{the cat}) \cdot p(\text{the} | \text{the cat chased}) \]

**Lexicon / vocabulary**

<table>
<thead>
<tr>
<th>type</th>
<th>spelling</th>
<th>embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>(w)</td>
<td>(\sigma(w))</td>
<td>(e(w))</td>
</tr>
<tr>
<td>1</td>
<td>the</td>
<td>[0.2, ..., 0.0]</td>
</tr>
<tr>
<td>2</td>
<td>cat</td>
<td>[0.4, ..., 0.5]</td>
</tr>
<tr>
<td>3</td>
<td>chased</td>
<td>[-0.1, ..., 0.2]</td>
</tr>
<tr>
<td>4</td>
<td>UNK</td>
<td>[0.3, ..., 0.1]</td>
</tr>
</tbody>
</table>

**Text generation with an RNN**

...but what is the word?

*Pure character-level model as the solution?*

...can’t we memorize it?

Ugh, spelling *the* again...
Our model: Spell once, summon anywhere

Our model: Spell once, summon anywhere

Our model: Spell once, summon anywhere
Our model: Spell once, summon anywhere

Known words only have to be spelled out once,
Our model: Spell once, summon anywhere

Known words only have to be **spelled out once**,
Our model: Spell once, summon anywhere

Known words only have to be **spelled out once**, 

![Diagram showing e(1), e(2), and e(3) with connections to t and h.]
Known words only have to be spelled out once,
Our model: Spell once, summon anywhere

Known words only have to be spelled out once,
Our model: Spell once, summon anywhere

Known words only have to be **spelled out once**,
Our model: Spell once, summon anywhere

Known words only have to be **spelled out once**,
Our model: Spell once, summon anywhere

Known words only have to be *spelled out once*,

\[ e(1) \rightarrow \sigma(1) \rightarrow \sigma(2) \rightarrow \sigma(3) \]

\[ \vec{h}_1 = \sigma(w_1) \]
\[ \vec{h}_2 = \sigma(w_2) \]
\[ \vec{h}_3 = \sigma(w_3) \]
\[ \vec{h}_4 = \sigma(w_4) \]
Our model: Spell once, summon anywhere

Known words only have to be spelled out once, and can then be summoned anywhere:
Our model: Spell once, summon anywhere

Known words only have to be spelled out once, and can then be summoned anywhere:
Our model: Spell once, summon anywhere

Known words only have to be spelled out once, and can then be summoned anywhere:
Our model: Spell once, summon anywhere

Known words only have to be spelled out once, and can then be summoned anywhere:
Our model: Spell once, summon anywhere

Known words only have to be spelled out once, and can then be summoned anywhere:

\[
\begin{align*}
    \vec{h}_1 & \xrightarrow{\text{RNN}} \vec{h}_2 \\
    w_1 & = e(1) \\
    w_2 & = e(2) \\
    \sigma(w_1) & \xrightarrow{\text{RNN}} \vec{h}_3 \\
    \sigma(w_2) & \xrightarrow{\text{RNN}} \vec{h}_4 \\
    \vec{h}_1 & = \text{the} \\
    \vec{h}_2 & = \text{chased} \\
    \vec{h}_3 & = \text{the} \\
    \vec{h}_4 & = \text{chased}
\end{align*}
\]
Our model: Spell once, summon anywhere

Known words only have to be spelled out once, and can then be summoned anywhere:
Our model: Spell once, summon anywhere

Known words only have to be **spelled out once**, and can then be **summoned anywhere**:
Our model: Spell once, summon anywhere

Known words only have to be spelled out once, and can then be summoned anywhere:

\[
\begin{align*}
&e(1) \\
\sigma(1) &
\\
e(2) &
\\
\sigma(2) &
\\
e(3) &
\\
\sigma(3) &
\end{align*}
\]

\[
\begin{align*}
\mu &
\\
\sigma(w_1) &
\\
\sigma(w_2) &
\\
\sigma(w_3) &
\end{align*}
\]

\[
\begin{align*}
\vec{h}_1 &\xrightarrow{\text{RNN}} \vec{h}_2 &\xrightarrow{\text{RNN}} \vec{h}_3
\end{align*}
\]

look up embeddings

look up spellings
Our model: Spell once, summon anywhere

Known words only have to be **spelled out once**, and can then be **summoned anywhere**:
Our model: Spell once, summon anywhere

Known words only have to be **spelled out once**, and can then be **summoned anywhere**:
Our model: Spell once, summon anywhere

Known words only have to be spelled out once, and can then be summoned anywhere:
Our model: Spell once, summon anywhere

Known words only have to be spelled out once, and can then be summoned anywhere:
Known words only have to be spelled out once, and can then be summoned anywhere.
Our model: Spell once, summon anywhere – the open-vocabulary case

Known words only have to be spelled out once, and can then be summoned anywhere.

\[ e^{(\text{UNK})} \]

\[ e^{(1)} \]

\[ e^{(2)} \]

\[ e^{(3)} \]

\[ w_1 = 1 \]

\[ w_2 = 2 \]

\[ w_4 = 1 \]

\[ \hat{h}_1 \]

\[ \hat{h}_2 \]

\[ \hat{h}_3 \]

\[ \hat{h}_4 \]

\[ \sigma(w_1) \]

\[ \sigma(w_2) \]

\[ \sigma(w_4) \]

\[ \sigma(\text{the}) \]

\[ \sigma(\text{cat}) \]

\[ \sigma(\text{caged}) \]

\[ \sigma(\text{the}) \]
Our model: Spell once, summon anywhere – the open-vocabulary case

Known words only have to be spelled out once, and can then be summoned anywhere.

\[
\begin{align*}
\vec{h}_1 &\rightarrow \vec{h}_2 \\
&\leftarrow w_1 = 1 \\
\vec{h}_2 &\rightarrow \vec{h}_3 \\
&\leftarrow w_2 = 2 \\
\vec{h}_3 &\rightarrow \vec{h}_4 \\
&\leftarrow w_3 = \text{UNK} \\
\vec{h}_4 &\rightarrow \text{the} \\
&\leftarrow w_4 = 1
\end{align*}
\]
Known words only have to be **spelled out once**, and can then be **summoned anywhere**. Unknown words are spelled out “on-demand” using the same character-level model.
Our model: Spell once, summon anywhere – the open-vocabulary case

Known words only have to be spelled out once, and can then be summoned anywhere. Unknown words are spelled out “on-demand” using the same character-level model.
Our model: Spell once, summon anywhere – the open-vocabulary case

Known words only have to be spelled out once, and can then be summoned anywhere. Unknown words are spelled out “on-demand” using the same character-level model.
Our model: Spell once, summon anywhere – the open-vocabulary case

Known words only have to be spelled out once, and can then be summoned anywhere. Unknown words are spelled out “on-demand” using the same character-level model.
Known words only have to be spelled out once, and can then be summoned anywhere. Unknown words are spelled out “on-demand” using the same character-level model.
Samples from the model

Sampled text from our model:

Following the death of Edward Mc-Cartney in **1060**, the new definition was transferred to the **WDIC** of **Fullett**.
Samples from the model

Sampled text from our model:

Following the death of Edward Mc-Cartney in 1060, the new definition was transferred to the WDIC of Fullett. novel word with contextually appropriate spelling
Samples from the model

Sampled text from our model:

Following the death of Edward McCartney in 1060, the new definition was transferred to the WDIC of Fullett.

novel word

with contextually appropriate spelling

known spelling $\sim$ novel spelling sampled from its embedding
Samples from the model:

Sampled text from our model:

Following the death of Edward Mc-Cartney in 1060, the new definition was transferred to the WDIC of Fullett.  

known spelling $\sim$ novel spelling sampled from its embedding

grounded $\sim$ stripped

novel word with contextually appropriate spelling
Samples from the model

Sampled text from our model:

Following the death of Edward McCartney in 1060, the new definition was transferred to the WDIC of Fullett. novel word

with contextually appropriate spelling

<table>
<thead>
<tr>
<th>known spelling</th>
<th>novel spelling sampled from its embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>grounded</td>
<td>stipped</td>
</tr>
<tr>
<td>differ</td>
<td>coronate</td>
</tr>
</tbody>
</table>
Samples from the model

Sampled text from our model:

Following the death of Edward Mc-Cartney in 1060, the new definition was transferred to the **WDIC** of **Fullett**.

---

<table>
<thead>
<tr>
<th>known spelling</th>
<th>novel spelling sampled from its embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>grounded</td>
<td><strong>stipped</strong></td>
</tr>
<tr>
<td>differ</td>
<td><strong>coronate</strong></td>
</tr>
<tr>
<td>Clive</td>
<td><strong>Dickey</strong></td>
</tr>
</tbody>
</table>

---

With **contextually appropriate spelling**

---
Samples from the model

Sampled text from our model:

Following the death of Edward McCartney in 1060, the new definition was transferred to the WDIC of Fullett. A novel word with contextually appropriate spelling...

<table>
<thead>
<tr>
<th>known spelling</th>
<th>novel spelling sampled from its embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>grounded</td>
<td>stipped</td>
</tr>
<tr>
<td>differ</td>
<td>coronate</td>
</tr>
<tr>
<td>Clive</td>
<td>Dickey</td>
</tr>
<tr>
<td>Southport</td>
<td>Strigger</td>
</tr>
</tbody>
</table>
Samples from the model

Sampled text from our model:

Following the death of Edward Mc-Cartney in 1060, the new definition was transferred to the WDIC of Fullett.

<table>
<thead>
<tr>
<th>known spelling</th>
<th>novel spelling sampled from its embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>grounded</td>
<td>stipped</td>
</tr>
<tr>
<td>differ</td>
<td>coronate</td>
</tr>
<tr>
<td>Clive</td>
<td>Dickey</td>
</tr>
<tr>
<td>Southport</td>
<td>Strigger</td>
</tr>
<tr>
<td>Carl</td>
<td>Wuly</td>
</tr>
</tbody>
</table>
Samples from the model:

Following the death of Edward McCartney in 1060, the new definition was transferred to the WDIC of Fullett. A novel word with contextually appropriate spelling.

<table>
<thead>
<tr>
<th>known spelling</th>
<th>novel spelling sampled from its embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>grounded</td>
<td>stipped</td>
</tr>
<tr>
<td>differ</td>
<td>cororate</td>
</tr>
<tr>
<td>Clive</td>
<td>Dickey</td>
</tr>
<tr>
<td>Southport</td>
<td>Strigger</td>
</tr>
<tr>
<td>Carl</td>
<td>Wuly</td>
</tr>
<tr>
<td>Chants</td>
<td>Tranquels</td>
</tr>
</tbody>
</table>
Samples from the model

Sampled text from our model:

Following the death of Edward McCartney in 1060, the new definition was transferred to the WDIC of Fullett. A novel word with contextually appropriate spelling.
Samples from the model

Sampled text from our model:

Following the death of Edward McCartney in 1060, the new denomination was transferred to the WDIC of Fullett. novel word with contextually appropriate spelling.

<table>
<thead>
<tr>
<th>known spelling</th>
<th>novel spelling sampled from its embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>grounded</td>
<td>stipped</td>
</tr>
<tr>
<td>differ</td>
<td>coronate</td>
</tr>
<tr>
<td>Clive</td>
<td>Dickey</td>
</tr>
<tr>
<td>Southport</td>
<td>Strigger</td>
</tr>
<tr>
<td>Carl</td>
<td>Wuly</td>
</tr>
<tr>
<td>Chants</td>
<td>Tranquels</td>
</tr>
<tr>
<td>valuables</td>
<td>migrations</td>
</tr>
</tbody>
</table>

So why is this a good way of modeling language?
The meaningful elements in any language—"words" in everyday parlance [...]— [...] are represented by [a] small stock of distinguishable sounds which are in themselves wholly meaningless. – Hockett, 1960
The meaningful elements in any language—"words" in everyday parlance — are represented by [a] small stock of distinguishable sounds which are in themselves wholly meaningless. – Hockett, 1960

“Meaningless” character composition should be separate from “meaningful” word composition!
The meaningful elements in any language—"words" in everyday parlance [...]—[...] are represented by [a] small stock of distinguishable sounds which are in themselves wholly meaningless. – Hockett, 1960

“Meaningless” character composition should be separate from “meaningful” word composition!

We should need a word’s spelling only to define it – not to later use it.
So? Why does this linguistics blurb matter?
So? Why does this linguistics blurb matter?

- Irregular words have uncommon spellings ...yet we use them like regular words!

\[ \text{children} \]
Duality of patterning $\leftrightarrow$ conditional independence!

So? Why does this linguistics blurb matter?

- Irregular words have uncommon spellings ...yet we use them like regular words!
- Function words have uncommon spellings ...yet we use them all the time without feeling weird!
So? Why does this linguistics blurb matter?

- Irregular words have uncommon spellings ...yet we use them like regular words!
- Function words have uncommon spellings ...yet we use them all the time without feeling weird!

Recall: We should need a word’s spelling only to define it – not to later use it.
Duality of patterning \(\leftrightarrow\) conditional independence!

So? Why does this linguistics blurb matter?

- Irregular words have uncommon spellings ...yet we use them like regular words!
- Function words have uncommon spellings ...yet we use them all the time without feeling weird!

Recall: We should need a word’s spelling only to define it – not to later use it.

\(\uparrow\) i.e. character-level models do it wrong!
So? Why does this linguistics blurb matter?

- Irregular words have uncommon spellings ...yet we use them like regular words!
- Function words have uncommon spellings ...yet we use them all the time without feeling weird!

Recall: We should need a word’s spelling only to *define* it – not to later *use* it.

↑ i.e. character-level models do it wrong!
...and they’re slow as hell...
So? Why does this linguistics blurb matter?

- Irregular words have uncommon spellings  
  ...yet we use them like regular words!
- Function words have uncommon spellings  
  ...yet we use them all the time without feeling weird!

Recall: We should need a word’s spelling only to *define* it – not to later *use* it.

↑ i.e. character-level models do it wrong!

...and they’re slow as hell...

💡 usage ⊥ spelling | embedding
How should a word’s embedding and its spelling be connected?
The arbitrariness of the sign → allowing for idiosyncracy

How should a word’s embedding and its spelling be connected?

*The connection between the signifier and the signified is arbitrary.*

– de Saussure, 1916, translated
The arbitrariness of the sign ⟷ allowing for idiosyncracy

How should a word’s embedding and its spelling be connected?

*The connection between the signifier and the signified is arbitrary.*

spelling \(\uparrow\) meaning

– de Saussure, 1916, translated
How should a word’s embedding and its spelling be connected?

*The connection between the signifier and the signified is arbitrary.*

Meaning is *not fully predictable* from spellings.
The arbitrariness of the sign → allowing for idiosyncracy

How should a word’s embedding and its spelling be connected?

*The connection between the signifier and the signified is arbitrary.*

spelling ↩️ meaning

– de Saussure, 1916, translated

Meaning is *not fully predictable* from spellings.

Example: neither *silly* nor *folly* is an adverb, even though they both end in -*ly*!
How should a word’s embedding and its spelling be connected?

*The connection between the signifier and the signified is arbitrary.*

- spelling ⇑
- meaning

– de Saussure, 1916, translated

Meaning is **not fully** predictable from spellings.

Example: neither *silly* nor *folly* is an adverb, even though they both end in -ly!

⇑ “construction” models like $e(caged) := \text{CNN}(\text{caged})$ ignore this!
The arbitrariness of the sign → allowing for idiosyncracy

How should a word’s embedding and its spelling be connected?

*The connection between the signifier and the signified is arbitrary.*

- spelling ↗
- meaning ↘

– de Saussure, 1916, translated

**Meaning is not fully predictable from spellings.**

Example: neither *silly* nor *folly* is an adverb, even though they both end in *-ly*!

“construction” models like $e(\text{caged}) := \text{CNN(c a g e d)}$ ignore this!

⇒ Allow any pairing a priori, but use spellings as prior / regularization!
The arbitrariness of the sign → allowing for idiosyncracy

How should a word’s embedding and its spelling be connected?

The connection between the signifier and the signified is arbitrary.

spelling ↗

meaning ↗

– de Saussure, 1916, translated

Meaning is **not fully** predictable from spellings.

Example: neither silly nor folly is an adverb, even though they both end in -ly!

“construction” models like $e(\text{caged}) := \text{CNN(\text{caged})}$ ignore this!

⇒ Allow any pairing a priori, but use spellings as prior / regularization!

Outliers (children, the, ...) may have idiosyncratic embeddings!
The arbitrariness of the sign → allowing for idiosyncracy

How should a word’s embedding and its spelling be connected?

*The connection between the signifier and the signified is arbitrary.*

spelling \( \uparrow \) meaning

– de Saussure, 1916, translated

Meaning is not **fully** predictable from spellings.

Example: neither *silly* nor *folly* is an adverb, even though they both end in *-ly*!

\( \uparrow \) “construction” models like \( e(caged) := \text{CNN}(\text{caged}) \) ignore this!

⇒ Allow any pairing a priori, but use spellings as prior / regularization!

Outliers (children, the, …) may have *idiosyncratic* embeddings!

💡 **regularize embeddings, don’t construct them** 💡
Recap: how does our model implement these ideas?

Embeddings and spellings are connected on the type level, ensuring conditional independence of usage and spelling while assigning positive probability to any pairing!
How do we evaluate open-vocabulary language models?

1. Report likelihood $p(\text{held-out text})$ as perplexity? (↓ lower is better)

→ no UNKing allowed!

→ we must predict every character of the text, regardless of vocabulary size

⇒ A tunable "vocabulary size" hyperparameter decides what is temporary-UNK.
How do we evaluate open-vocabulary language models?

**bits per character**

1. Report likelihood $p(\text{held-out text})$ as perplexity (↓ lower is better)
How do we evaluate open-vocabulary language models?

1. Report likelihood $p(\text{held-out text})$ as \textit{perplexity} (\downarrow lower is better)
2. \textcolor{orange}{\textbf{⚠️}} no UNKing allowed!
How do we evaluate open-vocabulary language models?

1. Report likelihood $p$ (held-out text) as perplexity (↓ lower is better)

2. ⚠️ no UNKing allowed!*
How do we evaluate open-vocabulary language models?

1. Report likelihood $p(\text{held-out text})$ as perplexity \(\downarrow\) lower is better
2. △ no UNKing allowed!*

* Yes, we call some words “UNK” temporarily, but we still generate them fully!
How do we evaluate open-vocabulary language models?

1. Report likelihood $p$ (held-out text) as perplexity (\(\downarrow\) lower is better)

2. \(\triangle\) no UNKing allowed!*
   → we must predict every character of the text, regardless of vocabulary size

* Yes, we call some words “UNK” temporarily, but we still generate them fully!
How do we evaluate open-vocabulary language models?

**bits per character**

1. Report likelihood $p$(held-out text) as perplexity (↓ lower is better)

2. **⚠️ no UNKing allowed!**
   - we must predict every character of the text, regardless of vocabulary size

* Yes, we call some words “UNK” *temporarily*, but we still *generate them fully*!

⇒ A tunable “vocabulary size” hyperparameter decides what is temporary-UNK.
Results

WikiText-2 (Merity et al., 2017)
2.5 million tokenized words from the English Wikipedia

← 1.8
1.4

...and plenty more baselines, ablations, datasets, and questions answered in the paper!
<table>
<thead>
<tr>
<th>WikiText-2 (Merity et al., 2017)</th>
<th>test</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5 million tokenized words from the English Wikipedia</td>
<td>1.775</td>
</tr>
<tr>
<td>← 1.8</td>
<td>1.4</td>
</tr>
<tr>
<td>the cat chased</td>
<td></td>
</tr>
<tr>
<td>BPE:</td>
<td></td>
</tr>
<tr>
<td>Spell Once, Summon Anywhere</td>
<td>4.01</td>
</tr>
<tr>
<td>our full model:</td>
<td>4.00</td>
</tr>
</tbody>
</table>

...and plenty more baselines, ablations, datasets, and questions answered in the paper!
## Results

<table>
<thead>
<tr>
<th>Model</th>
<th>Loss</th>
<th>Scale Factor</th>
<th>Test Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WikiText-2</strong> (Merity et al., 2017)</td>
<td></td>
<td></td>
<td>1.8</td>
</tr>
<tr>
<td>2.5 million tokenized words from the English Wikipedia</td>
<td></td>
<td></td>
<td>1.4</td>
</tr>
<tr>
<td>t h e c a t c h a s e d</td>
<td>1.775</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HCLM + cache</td>
<td></td>
<td></td>
<td>1.500</td>
</tr>
<tr>
<td>previous SOTA (Kawakami et al., 2017)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spell Once, Summon Anywhere</td>
<td>4.01</td>
<td></td>
<td>1.64</td>
</tr>
</tbody>
</table>
Results

WikiText-2 (Merity et al., 2017)
2.5 million tokenized words from the English Wikipedia

← 1.8

\[ \text{test} \]

\[ 1.4 \parallel \]

\[ 1.775 \]

HCLM + cache

\[ \text{previous SOTA (Kawakami et al., 2017)} \]

\[ 1.500 \]

BPE:

\[ 1.468 \]

"the cat chased"

"the ca@ @ t ch@ @ sed"
## Results

### WikiText-2 (Merity et al., 2017)

2.5 million tokenized words from the English Wikipedia

<table>
<thead>
<tr>
<th>Model</th>
<th>Loss</th>
<th>Test Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>← 1.8</td>
<td>1.4</td>
<td></td>
</tr>
<tr>
<td>the cat chased</td>
<td>1.775</td>
<td></td>
</tr>
<tr>
<td>HCLM + cache</td>
<td>1.500</td>
<td></td>
</tr>
<tr>
<td>previous SOTA (Kawakami et al., 2017)</td>
<td>1.468</td>
<td></td>
</tr>
</tbody>
</table>

### BPE:

<table>
<thead>
<tr>
<th>Model</th>
<th>Loss</th>
<th>Test Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>the cat chased</td>
<td>1.468</td>
<td></td>
</tr>
</tbody>
</table>
## Results

<table>
<thead>
<tr>
<th>Method</th>
<th>The cat chased</th>
<th>Test Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>WikiText-2 (Merity et al., 2017)</td>
<td>1.8</td>
<td>1.4 (|)</td>
</tr>
<tr>
<td>HCLM + cache (previous SOTA (Kawakami et al., 2017))</td>
<td>1.775</td>
<td>1.500</td>
</tr>
<tr>
<td>BPE:</td>
<td>1.468</td>
<td></td>
</tr>
<tr>
<td>our full model: Spell Once, Summon Anywhere</td>
<td>1.455</td>
<td></td>
</tr>
</tbody>
</table>
## Results

<table>
<thead>
<tr>
<th>Model</th>
<th>Novel words</th>
<th>Rare words</th>
<th>Frequent words</th>
<th>Test all</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WikiText-2</strong> (Merity et al., 2017)</td>
<td>3.89</td>
<td>2.08</td>
<td>1.38</td>
<td>1.775</td>
</tr>
<tr>
<td><strong>HCLM + cache</strong></td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>1.500</td>
</tr>
<tr>
<td><strong>BPE:</strong></td>
<td>4.01</td>
<td>1.70</td>
<td>1.08</td>
<td>1.468</td>
</tr>
<tr>
<td><em>our full model:</em> Spell Once, Summon Anywhere</td>
<td>4.00</td>
<td><strong>1.64</strong></td>
<td>1.10</td>
<td><strong>1.455</strong></td>
</tr>
</tbody>
</table>
## Results

WikiText-2 (Merity et al., 2017)

2.5 million tokenized words from the English Wikipedia

<table>
<thead>
<tr>
<th></th>
<th>novel words</th>
<th>rare words</th>
<th>frequent words</th>
<th>test all</th>
</tr>
</thead>
<tbody>
<tr>
<td>the cat chased</td>
<td>3.89</td>
<td>2.08</td>
<td>1.38</td>
<td>1.775</td>
</tr>
<tr>
<td>HCLM + cache</td>
<td></td>
<td></td>
<td></td>
<td>1.500</td>
</tr>
<tr>
<td>previous SOTA (Kawakami et al., 2017)</td>
<td></td>
<td></td>
<td></td>
<td>1.468</td>
</tr>
<tr>
<td>BPE:</td>
<td>4.01</td>
<td>1.70</td>
<td>1.08</td>
<td>1.455</td>
</tr>
</tbody>
</table>

**our full model:** Spell Once, Summon Anywhere

4.00 1.64 1.10 1.455

...and plenty more baselines, ablations, datasets, and questions answered in the paper!
1. think about language before you model:
   - usage ⊥ spelling | embedding
   - regularize embeddings, don’t construct them
1. think about language before you model:
   - usage ⊥ spelling | embedding
   - regularize embeddings, don’t construct them

2. simple and criminally underused baselines can beat fancy but bad models
   - model strings by segments?
1. think about language before you model:
   - usage ⊥ spelling | embedding
   - regularize embeddings, don’t construct them

2. simple and criminally underused baselines can beat fancy but bad models
   - model strings by segments?

3. open-vocabulary language modeling is an exciting task!
Spell Once, Summon Anywhere:
A Two-Level Open-Vocabulary Language Model
AAAI 2019 Technical Track

Sebastian J. Mielke  and  Jason Eisner
sjmielke@jhu.edu, jason@cs.jhu.edu
Department of Computer Science, Johns Hopkins University, Baltimore, MD, USA

Check out our poster yesterday!

Twitter: sjmielke
E-Mail: sjmielke@jhu.edu