Neural Machine Translation:
Attention Model

1/17/2017
Today’s Focus

- Neural Machine Translation with Attention
- Specific focus on what you need to know to understand NMT with Attention, as implemented in the Nematus toolkit (used in Edinburgh’s WMT’16 submission)
If you want to check individual line numbers referenced in these slides, look at this version of the code (most recent at the time of writing):

https://github.com/rsennrich/nematus/tree/9b08048e278db46ede288bcca928d176811ee2c2
Topics

• GRU
• Basic RNN Encoder-Decoder for NMT
• Attention model
• Other topics
  – Search
  – Ensembling
  – Synthetic data
  – Additional linguistic features
  – Byte-Pair Encoding review
Gated Recurrent Unit (GRU)

• *Reset gate*  
  – Determine the tradeoff between previous hidden state and new input.

• *Proposed hidden state*  
  – Combine previous hidden state with new input, according to reset gate.

• *Update gate*  
  – How much should we update the hidden state? Use the previous hidden state? This proposed one? Some interpolation?
GRU

\[ z_t = \sigma \left( W_z \cdot [h_{t-1}, x_t] \right) \]

\[ r_t = \sigma \left( W_r \cdot [h_{t-1}, x_t] \right) \]

\[ \tilde{h}_t = \tanh \left( W \cdot [r_t \cdot h_{t-1}, x_t] \right) \]

\[ h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h}_t \]
GRU

\[ z_t = \sigma (W_z \cdot [h_{t-1}, x_t]) \]
\[ r_t = \sigma (W_r \cdot [h_{t-1}, x_t]) \]
\[ \tilde{h}_t = \tanh (W \cdot [r_t \ast h_{t-1}, x_t]) \]
\[ h_t = (1 - z_t) \ast h_{t-1} + z_t \ast \tilde{h}_t \]

**Reset gate**

Images from: [http://colah.github.io/posts/2015-08-Understanding-LSTMs/](http://colah.github.io/posts/2015-08-Understanding-LSTMs/)
GRU

\[ z_t = \sigma (W_z \cdot [h_{t-1}, x_t]) \]

\[ r_t = \sigma (W_r \cdot [h_{t-1}, x_t]) \]

\[ \tilde{h}_t = \tanh (W \cdot [r_t \times h_{t-1}, x_t]) \]

\[ h_t = (1 - z_t) \times h_{t-1} + z_t \times \tilde{h}_t \]

Reset gate

**Proposed Hidden State**

Images from: [http://colah.github.io/posts/2015-08-Understanding-LSTMs/](http://colah.github.io/posts/2015-08-Understanding-LSTMs/)
GRU

\[ z_t = \sigma \left( W_z \cdot [h_{t-1}, x_t] \right) \]
\[ r_t = \sigma \left( W_r \cdot [h_{t-1}, x_t] \right) \]
\[ \tilde{h}_t = \tanh \left( W \cdot [r_t \ast h_{t-1}, x_t] \right) \]
\[ h_t = (1 - z_t) \ast h_{t-1} + z_t \ast \tilde{h}_t \]

**Update gate**

**Reset gate**

**Proposed Hidden State**

Images from: [http://colah.github.io/posts/2015-08-Understanding-LSTMs/](http://colah.github.io/posts/2015-08-Understanding-LSTMs/)
GRU

Update gate

Reset gate

Proposed Hidden State

Hidden State

\[ z_t = \sigma (W_z \cdot [h_{t-1}, x_t]) \]

\[ r_t = \sigma (W_r \cdot [h_{t-1}, x_t]) \]

\[ \tilde{h}_t = \tanh (W \cdot [r_t \ast h_{t-1}, x_t]) \]

\[ h_t = (1 - z_t) \ast h_{t-1} + z_t \ast \tilde{h}_t \]
**GRU**

**Update gate**

\[ z_t = \sigma (W_z \cdot [h_{t-1}, x_t]) \]

**Reset gate**

\[ r_t = \sigma (W_r \cdot [h_{t-1}, x_t]) \]

**Proposed Hidden State**

\[ \tilde{h}_t = \tanh (W \cdot [r_t \ast h_{t-1}, x_t]) \]

**Hidden State**

\[ h_t = (1 - z_t) \ast h_{t-1} + z_t \ast \tilde{h}_t \]

Update tradeoff between:

- previous hidden state
- proposed hidden state

Images from: [http://colah.github.io/posts/2015-08-Understanding-LSTMs/](http://colah.github.io/posts/2015-08-Understanding-LSTMs/)
**GRU**

**Update gate**

\[ z_t = \sigma (W_z \cdot [h_{t-1}, x_t]) \]

**Reset gate**

\[ r_t = \sigma (W_r \cdot [h_{t-1}, x_t]) \]

**Proposed Hidden State**

\[ \tilde{h}_t = \tanh (W \cdot [r_t \ast h_{t-1}, x_t]) \]

**Hidden State**

\[ h_t = (1 - z_t) \ast h_{t-1} + z_t \ast \tilde{h}_t \]

Update tradeoff between:

previous hidden state and proposed hidden state
GRU

**Update gate**

$z_t = \sigma (W_z \cdot [h_{t-1}, x_t])$

**Reset gate**

$r_t = \sigma (W_r \cdot [h_{t-1}, x_t])$

**Proposed Hidden State**

$\tilde{h}_t = \tanh (W \cdot [r_t \ast h_{t-1}, x_t])$

**Hidden State**

$h_t = (1 - z_t) \ast h_{t-1} + z_t \ast \tilde{h}_t$

If $z=1$:
- ignore previous state, use proposed state

If $z=0$:
- ignore proposed state, use previous state

Images from: [http://colah.github.io/posts/2015-08-Understanding-LSTMs/](http://colah.github.io/posts/2015-08-Understanding-LSTMs/)
GRU

\[ r_j = \sigma \left( [W_r x]_j + [U_r h_{(t-1)}]_j \right) \]

\[ \tilde{h}_j^{(t)} = \phi \left( [W x]_j + [U (r \odot h_{(t-1)})]_j \right) \]

\[ z_j = \sigma \left( [W_z x]_j + [U_z h_{(t-1)}]_j \right) \]

\[ h_j^{(t)} = z_j h_j^{(t-1)} + (1 - z_j) \tilde{h}_j^{(t)} \]

GRU

Notation: $t$ fixed; $j$-th element in the vector

“each hidden unit will learn to capture dependencies over different time scales”

$$r_j = \sigma \left( [W_r x]_j + [U_r h^{(t-1)}]_j \right)$$

$$\tilde{h}^{(t)}_j = \phi \left( [W x]_j + [U (r \odot h^{(t-1)})]_j \right)$$

$$z_j = \sigma \left( [W_z x]_j + [U_z h^{(t-1)}]_j \right)$$

$$h^{(t)}_j = z_j h^{(t-1)}_j + (1 - z_j) \tilde{h}^{(t)}_j$$

GRU: Nematus Code

nematus/layers.py, lines 71-98, 101-154

(https://github.com/rsennrich/nematus/blob/9b08048e278db46ede288bcca928d176811ee2c2/nematus/layers.py)
Gated Recurrent Unit (GRU)


Chris Olah. 2015. Understanding LSTMs. (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)
Basic RNN Encoder-Decoder

Notation note:

In general, neural models tend to be drawn with the input at the bottom and output at the top.

In the rest of this presentation, I will be drawing them with the input at the top.

(Flipping them over the horizontal axis will result in the standard version.)
Ein souveräner Auftakt in der Saison
One-hot representation

Input
Continuous representation (word embeddings)
As visual shorthand for the input (one-hot into continuous), we’ll just use this to save space:
Ein souveräner Auftakt in der Saison <EOS>
\[ \mathbf{x} = (x_1, \ldots, x_T) \]

(here, \( T = 7 \))
RNN Encoder

\[ h_{(t)} = f(h_{(t-1)} , x_t) \]
RNN Encoder

\[ h_{(t)} = f(h_{(t-1)} , x_t) \]

Let’s assume \( f \) is a GRU (or LSTM).
The last hidden state of the RNN, \( c \) (formerly \( h_7 \)), contains a *summary* of the input sentence \( x \).
RNN Decoder
RNN Decoder

\[ h(t) = f(h(t-1), y(t-1), c) \]
RNN Decoder

\[
h(t) = f(h(t-1), y(t-1), c)
\]
RNN Decoder

\[ h(t) = f(h(t-1), y(t-1), c) \]
RNN Decoder

\[ h_{(t)} = f(h_{(t-1)}, y_{(t-1)}, c) \]
RNN Decoder

\[ h(t) = f(h(t-1), y(t-1), c) \]
RNN Decoder: Zoomed In

\[
P(y_{(t)}|y_{(t-1)}, \ldots, y_1, c) = g(h_{(t)}, y_{(t-1)}, c)
\]
Quick Note: 
$y_{(t-1)}$ refers to a representation of the single word predicted.

$$h_{(t)} = f(h_{(t-1)}, y_{(t-1)}, c)$$
RNN Encoder-Decoder Math

Input: \[ x = (x_1, \ldots, x_T) \]

Encoder:
\[
\begin{align*}
    h(t) &= f(h_{(t-1)}, x_t) \\
    c &= h_{(T)}
\end{align*}
\]

Decoder:
\[
\begin{align*}
    h(t) &= f(h_{(t-1)}, y_{(t-1)}, c) \\
    P(y_{(t)}|y_{(t-1)}, \ldots, y_1, c) &= g(h_{(t)}, y_{(t-1)}, c)
\end{align*}
\]

Output: \[ y = (y_1, \ldots, y_{T'}) \]
Training

Train to maximize the log-likelihood of the corpus:

$$\underset{\theta}{\text{max}} \, \frac{1}{N} \sum_{n=1}^{N} \log p_{\theta}(y_n \mid x_n)$$
A Few Final Details...

• Start with a <BOS> token (not shown before).
• Translate until the <EOS> token is generated (or a max length is reached).
• This lacks:
  – Explicit alignments (to be discussed next)
  – Coverage vector
Basic RNN Encoder-Decoder


Attention

Intuition

• Summary vector is a bottleneck, especially for long sentences
• Want to introduce some form of alignment
Attention

Soft attention, visualized.

High-Level Walk-Through
Ein souveräner Auftakt in der Saison
Ein souveräner Auftakt in der Saison.
(Quite a bit of detail between the input and the probability distribution over the vocabulary is omitted from this drawing, but will be discussed in the following slides.)
Ein souveräner Auftakt in der Saison.

Bidirectional RNN
Attention Mechanism
Ein souveräner Auftakt der Saison.

---

**Distribution Over Vocabulary**

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0.6</td>
</tr>
<tr>
<td>an</td>
<td>0.3</td>
</tr>
<tr>
<td>the</td>
<td>0.05</td>
</tr>
<tr>
<td>that</td>
<td>0.01</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>halloween</td>
<td>1e-9</td>
</tr>
<tr>
<td>trilobite</td>
<td>1e-10</td>
</tr>
</tbody>
</table>
Ein souveräner Auftakt in der souveränen Saison.

Condition on Previous Word
Ein souveräner Auftakt in der Saison.
New Distribution, Conditioned on Previous Selection
New Distribution, Conditioned on Previous Selection

Highest probability word: “confident”
New Distribution, Conditioned on Previous Selection

Highest probability word: “confident”
A confident start in the season
Attention: Bidirectional RNN

- Let hidden states contain more location-sensitive information
- Maintain information about the full sentence in the hidden states
RNN Encoder: Forward

\[ \hat{h}_{(i)} = f(\hat{h}_{(i-1)}, x_i) \]
RNN Encoder: Backward

\[
\hat{h}_i = f(\hat{h}_{(i-1)} , x_i)
\]
Concatenated

\[ h_{(i)} = \text{concat}(\vec{h}_{(i)}, \vec{h}_{(i)}) \]
RNN Encoder: Concatenated

• By concatenating a forward RNN and a backward RNN, we have a bidirectional RNN
• $f$ is a nonlinear function, likely a GRU or LSTM
• Each $h_i$ contains information about the current word $x_i$, and its context (in both directions)
• Intuitively, we may expect that $h_i$ will be particularly useful when translating $x_i$, but it may be useful for translating other words too
The “attention” or “soft alignments” will determine how much each hidden state (“annotation”) influences the generation of each word in the translation.
Attention Mechanism

Equations from Bahdanau et al

Attention Mechanism

- Context vectors
- Hidden states
- $\alpha$
Each time we generate a word $y_i$, we’ll create a context vector, $c_i$.

- $c_i$ is a weighted sum of the hidden states $h_j$

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

- (we’ll look at how to compute $\alpha$ shortly)
Context Vectors

- Initialization: context vector $c_0$ is the unweighted average of $h_j$ for all $j$
Context Vectors

- This differs from the earlier version we defined, which had a single $c$ for the whole input sentence.
Hidden States

\[ s_i = f(s_{i-1}, y_{i-1}, c_i) \]

\( s_i \) is an RNN hidden state for time \( i \)
\[ e_{ij} = a(s_{(i-1)}, h_j) \]

\( a \) is a feed-forward neural net, our alignment model

Think of it as checking how similar inputs (around \( j \)) are to outputs (around \( i \)).
\[ e_{ij} = a(s_{(i-1)}, h_j) \]

\[ \alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \]

(softmax)
Generating $y_i$

\[ P(y_{(i)} | y_{(i-1)}, \ldots, y_1, x) = g(y_{(i-1)}, s_{(i)}, c_i) \]
Attention Model Math

Encoding:

\( \mathbf{\hat{h}}_{(i)} = f(\mathbf{\hat{h}}_{(i-1)}, x_i) \)

\( \mathbf{\hat{h}}_{(i-1)} = f(\mathbf{\hat{h}}_{(i)}, x_i) \)

\( h_{(i)} = concat(\mathbf{\hat{h}}_{(i)}, \mathbf{\hat{h}}_{(i)}) \)
Attention Model Math

Attention:

\[ e_{ij} = a(s_{i-1}, h_j) \]

\[ \alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{Tx} \exp(e_{ik})} \]

\[ c_i = \sum_{j=1}^{Tx} \alpha_{ij} h_j \]

\[ s_i = f(s_{i-1}, y_{i-1}, c_i) \]
Attention Model Math

Decoding:

\[ P(y_{(i)} | y_{(i-1)}, \ldots, y_1, x) = g(y_{(i-1)}, s_{(i)}, c_i) \]
A confident start in the season
Attention Code

Implemented in *gru_cond*

nematus/layers.py, lines 171-239, 242-377

(https://github.com/rsennrich/nematus/blob/9b08048e278db46ede288bcca928d176811ee2c2/nematus/layers.py)
Attention


Other Useful Things To Know
Search

• In the examples, I’ve shown beam size of 1
  – Either sample from the probability distribution, or greedily choose the 1 best
  – Continue until reaching <eos> (or some max)

• Can also do beam search
  – Keep K hypotheses with highest probability
  – Expand each one at each step
Search

• For neural models, small beam sizes
• Beam size $\sim 12$ seems to work best, but:
  – most improvement seems to come even with just beam size $\sim 2$
  – it’s a bit slower
Ensembling

• Average the probability distributions of different models in order to generate words in translation
• Models must have same output vocab and factorization of $y$
• Can ensemble different training runs, different initializations, or ensemble using checkpoints from one run
Synthetic Training Data

• Dummy sentences
  – Given large monolingual target-language corpus.
  – “Translate” empty source sentences into target monolingual sentences.
  – Freeze encoder/attention parameters.

Synthetic Training Data

• **Back-translation**
  – Given a large target-language monolingual corpus, use an MT system to “back-translate” it into the source language.
  – Train on both human-translated parallel text and this MT-generated parallel text.

Additional Linguistic Features

- Add additional linguistic features to input
- Recall that vocab items are represented as lower-dimensional vectors of continuous values; do the same with additional features (“factored NMT”)

Problem: Vocabulary

- Neural models tend to have vocabulary sizes <50,000
- Want to be able to handle rare words
- May want to be able to generate novel words (for example, in morphologically complex languages)
Byte-Pair Encoding (BPE)

<table>
<thead>
<tr>
<th>word</th>
<th>freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘low’</td>
<td>5</td>
</tr>
<tr>
<td>‘lower’</td>
<td>2</td>
</tr>
<tr>
<td>‘newest’</td>
<td>6</td>
</tr>
<tr>
<td>‘widest’</td>
<td>3</td>
</tr>
</tbody>
</table>

- Treat each word as a sequence of character tokens (plus word-end token)
- Collect into dictionary (efficiency)
- Given dictionary and desired number of merges, merge most frequent pairs into symbols
- Return new character + symbol vocabulary
- Example from Sennrich, Birch, Junczys-Dowmunt
  (http://homepages.inf.ed.ac.uk/rsennric/amta2016-tutorial.pdf)
## Byte-Pair Encoding (BPE)

<table>
<thead>
<tr>
<th>word</th>
<th>freq</th>
<th>freq</th>
<th>symbol pair</th>
<th>new symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘l o w&lt;/w&gt;’</td>
<td>5</td>
<td>9</td>
<td>(‘e’, ‘s’)</td>
<td>‘es’</td>
</tr>
<tr>
<td>‘l o w e r&lt;/w&gt;’</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>‘n e w es t&lt;/w&gt;’</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>‘w i d es t&lt;/w&gt;’</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- ‘l o w</w>’ freq: 5
- ‘l o w e r</w>’ freq: 2
- ‘n e w es t</w>’ freq: 6
- ‘w i d es t</w>’ freq: 3

- Symbol pair ‘e’, ‘s’ -> new symbol ‘es’
## Byte-Pair Encoding (BPE)

<table>
<thead>
<tr>
<th>word</th>
<th>freq</th>
<th>freq</th>
<th>symbol pair</th>
<th>new symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘l o w &lt;/w&gt;’</td>
<td>5</td>
<td>9</td>
<td>(‘e’, ‘s’)</td>
<td>‘es’</td>
</tr>
<tr>
<td>‘l o w e r &lt;/w&gt;’</td>
<td>2</td>
<td>9</td>
<td>(‘es’, ‘t’)</td>
<td>‘st’</td>
</tr>
<tr>
<td>‘n e w est &lt;/w&gt;’</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>‘w i d est &lt;/w&gt;’</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Byte-Pair Encoding (BPE)

<table>
<thead>
<tr>
<th>word</th>
<th>freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘l o w &lt;/w&gt;’</td>
<td>5</td>
</tr>
<tr>
<td>‘l o w e r &lt;/w&gt;’</td>
<td>2</td>
</tr>
<tr>
<td>‘n e w est&lt;/w&gt;’</td>
<td>6</td>
</tr>
<tr>
<td>‘w i d est&lt;/w&gt;’</td>
<td>3</td>
</tr>
</tbody>
</table>

### Symbol Pairs

<table>
<thead>
<tr>
<th>freq</th>
<th>symbol pair</th>
<th>new symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>(‘e’, ‘s’)</td>
<td>‘es’</td>
</tr>
<tr>
<td>9</td>
<td>(‘es’,’t’)</td>
<td>‘est’</td>
</tr>
<tr>
<td>9</td>
<td>(‘est’,’&lt;/w&gt;’)</td>
<td>‘est&lt;/w&gt;’</td>
</tr>
</tbody>
</table>
## Byte-Pair Encoding (BPE)

<table>
<thead>
<tr>
<th>word</th>
<th>freq</th>
<th>freq</th>
<th>symbol pair</th>
<th>new symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>'lo w &lt;/w&gt;'</td>
<td>5</td>
<td>9</td>
<td>('e', 's')</td>
<td>'es'</td>
</tr>
<tr>
<td>'lo w e r &lt;/w&gt;'</td>
<td>2</td>
<td>9</td>
<td>('es', 't')</td>
<td>'est'</td>
</tr>
<tr>
<td>'n e w est&lt;/w&gt;'</td>
<td>6</td>
<td>9</td>
<td>('est', '&lt;/w&gt;')</td>
<td>'est&lt;/w&gt;'</td>
</tr>
<tr>
<td>'w i d est&lt;/w&gt;'</td>
<td>3</td>
<td>7</td>
<td>('l', 'o')</td>
<td>'lo'</td>
</tr>
</tbody>
</table>
# Byte-Pair Encoding (BPE)

<table>
<thead>
<tr>
<th>word</th>
<th>freq</th>
<th>symbol pair</th>
<th>new symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘low &lt;/w&gt;’</td>
<td>5</td>
<td>(‘e’, ‘s’)</td>
<td>→ ‘es’</td>
</tr>
<tr>
<td>‘low er &lt;/w&gt;’</td>
<td>2</td>
<td>(‘es’, ‘t’)</td>
<td>→ ‘est’</td>
</tr>
<tr>
<td>‘new est&lt;/w&gt;’</td>
<td>6</td>
<td>(‘est’, ‘&lt;/w&gt;’)</td>
<td>→ ‘est&lt;/w&gt;’</td>
</tr>
<tr>
<td>‘wid est&lt;/w&gt;’</td>
<td>3</td>
<td>(‘l’, ‘o’)</td>
<td>→ ‘lo’</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(‘lo’, ‘w’)</td>
<td>→ ‘low’</td>
</tr>
</tbody>
</table>

...
Byte-Pair Encoding (BPE)

<table>
<thead>
<tr>
<th>word</th>
<th>freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘low &lt;/w&gt;’</td>
<td>5</td>
</tr>
<tr>
<td>‘low e r &lt;/w&gt;’</td>
<td>2</td>
</tr>
<tr>
<td>‘n e w est&lt;/w&gt;’</td>
<td>6</td>
</tr>
<tr>
<td>‘w i d est&lt;/w&gt;’</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>freq</th>
<th>symbol pair</th>
<th>new symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>(‘e’, ‘s’)</td>
<td>→ ‘es’</td>
</tr>
<tr>
<td>9</td>
<td>(‘es’, ’t’)</td>
<td>→ ‘est’</td>
</tr>
<tr>
<td>9</td>
<td>(‘est’,’&lt;/w&gt;’)</td>
<td>→ ‘est&lt;/w&gt;’</td>
</tr>
<tr>
<td>7</td>
<td>(‘l’,’o’)</td>
<td>→ ‘lo’</td>
</tr>
<tr>
<td>7</td>
<td>(‘lo’,’w’)</td>
<td>→ ‘low’</td>
</tr>
</tbody>
</table>

...
## Byte-Pair Encoding (BPE)

<table>
<thead>
<tr>
<th>word</th>
<th>freq</th>
<th>freq</th>
<th>symbol pair</th>
<th>new symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘low &lt;/w&gt;’</td>
<td>5</td>
<td>9</td>
<td>(‘e’, ‘s’)</td>
<td>‘es’</td>
</tr>
<tr>
<td>‘low e r &lt;/w&gt;’</td>
<td>2</td>
<td>9</td>
<td>(‘es’, ’t’)</td>
<td>‘est’</td>
</tr>
<tr>
<td>‘n e w est&lt;/w&gt;’</td>
<td>6</td>
<td>9</td>
<td>(‘est’, ’&lt;/w&gt;’)</td>
<td>‘est&lt;/w&gt;’</td>
</tr>
<tr>
<td>‘w i d est&lt;/w&gt;’</td>
<td>3</td>
<td>7</td>
<td>(‘l’, ’o’)</td>
<td>‘lo’</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>(‘lo’, ’w’)</td>
<td>‘low’</td>
</tr>
</tbody>
</table>

... 

\[
\text{‘l o w e s t </w>’} \rightarrow
\]
## Byte-Pair Encoding (BPE)

<table>
<thead>
<tr>
<th>word</th>
<th>freq</th>
<th>freq</th>
<th>symbol pair</th>
<th>new symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘low &lt;/w&gt;’</td>
<td>5</td>
<td>9</td>
<td>(‘e’, ‘s’)</td>
<td>‘es’</td>
</tr>
<tr>
<td>‘low e r &lt;/w&gt;’</td>
<td>2</td>
<td>9</td>
<td>(‘es’, ‘t’)</td>
<td>‘est’</td>
</tr>
<tr>
<td>‘n e w est&lt;/w&gt;’</td>
<td>6</td>
<td>9</td>
<td>(‘est’, ’&lt;/w&gt;’)</td>
<td>‘est&lt;/w&gt;’</td>
</tr>
<tr>
<td>‘w i d est&lt;/w&gt;’</td>
<td>3</td>
<td>7</td>
<td>(‘l’, ‘o’)</td>
<td>‘lo’</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>(‘l’, ‘o’)</td>
<td>‘lo’</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

`‘l o w e s t </w>’` → `‘l o w es t </w>’`
## Byte-Pair Encoding (BPE)

<table>
<thead>
<tr>
<th>word</th>
<th>freq</th>
<th>freq</th>
<th>symbol pair</th>
<th>new symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘low &lt;/w&gt;’</td>
<td>5</td>
<td>9</td>
<td>(‘e’, ‘s’)</td>
<td>→ ‘es’</td>
</tr>
<tr>
<td>‘low e r &lt;/w&gt;’</td>
<td>2</td>
<td>9</td>
<td>(‘es’, ‘t’)</td>
<td>→ ‘est’</td>
</tr>
<tr>
<td>‘n e w est&lt;/w&gt;’</td>
<td>6</td>
<td>9</td>
<td>(‘est’, &lt;/w&gt;’)</td>
<td>→ ‘est&lt;/w&gt;’</td>
</tr>
<tr>
<td>‘w i d est&lt;/w&gt;’</td>
<td>3</td>
<td>7</td>
<td>(‘l’, ‘o’)</td>
<td>→ ‘lo’</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>(‘lo’, ‘w’)</td>
<td>→ ‘low’</td>
</tr>
</tbody>
</table>

...  

‘l o w es t </w>’ → ‘l o w est </w>’
# Byte-Pair Encoding (BPE)

<table>
<thead>
<tr>
<th>word</th>
<th>freq</th>
<th>freq</th>
<th>symbol pair</th>
<th>new symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘low &lt;/w&gt;’</td>
<td>5</td>
<td>9</td>
<td>(‘e’, ‘s’)</td>
<td>‘es’</td>
</tr>
<tr>
<td>‘low e r &lt;/w&gt;’</td>
<td>2</td>
<td>9</td>
<td>(‘es’, ’t’)</td>
<td>‘est’</td>
</tr>
<tr>
<td>‘n e w est&lt;/w&gt;’</td>
<td>6</td>
<td>9</td>
<td>(‘est’, ’&lt;/w&gt;’)</td>
<td>‘est&lt;/w&gt;’</td>
</tr>
<tr>
<td>‘w i d est&lt;/w&gt;’</td>
<td>3</td>
<td>7</td>
<td>(‘l’, ’o’)</td>
<td>‘lo’</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>(‘lo’, ’w’)</td>
<td>‘low’</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ \text{‘low est </w>’} \quad \rightarrow \quad \text{‘low est</w>’} \]
### Byte-Pair Encoding (BPE)

<table>
<thead>
<tr>
<th>word</th>
<th>freq</th>
<th>freq</th>
<th>symbol pair</th>
<th>new symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘low &lt;/w&gt;’</td>
<td>5</td>
<td>9</td>
<td>(‘e’, ‘s’)</td>
<td>‘es’</td>
</tr>
<tr>
<td>‘low e r &lt;/w&gt;’</td>
<td>2</td>
<td>9</td>
<td>(‘es’, ‘t’)</td>
<td>‘est’</td>
</tr>
<tr>
<td>‘n e w est&lt;/w&gt;’</td>
<td>6</td>
<td>9</td>
<td>(‘est’, &lt;/w&gt;’)</td>
<td>‘est&lt;/w&gt;’</td>
</tr>
<tr>
<td>‘w i d est&lt;/w&gt;’</td>
<td>3</td>
<td>7</td>
<td>(‘l’, ‘o’)</td>
<td>‘lo’</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>(‘lo’, ‘w’)</td>
<td>‘low’</td>
</tr>
</tbody>
</table>

...  

‘l o w est</w>’  →  ‘lo w est</w>’
# Byte-Pair Encoding (BPE)

<table>
<thead>
<tr>
<th>word</th>
<th>freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘low &lt;/w&gt;’</td>
<td>5</td>
</tr>
<tr>
<td>‘low e r &lt;/w&gt;’</td>
<td>2</td>
</tr>
<tr>
<td>‘n e w est &lt;/w&gt;’</td>
<td>6</td>
</tr>
<tr>
<td>‘w i d est &lt;/w&gt;’</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>freq</th>
<th>symbol pair</th>
<th>new symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>(‘e’, ‘s’)</td>
<td>‘es’</td>
</tr>
<tr>
<td>9</td>
<td>(‘es’, ‘t’)</td>
<td>‘est’</td>
</tr>
<tr>
<td>9</td>
<td>(‘es’, '&lt;/w&gt;')</td>
<td>‘est&lt;/w&gt;’</td>
</tr>
<tr>
<td>7</td>
<td>(‘l’, ‘o’)</td>
<td>‘lo’</td>
</tr>
<tr>
<td>7</td>
<td>(‘lo’, ‘w’)</td>
<td>‘low’</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

‘lo w est </w>’ → ‘low est </w>’
Alternatives

• Character-Level Models
  – Costa-Jussà and Fonollosa, ACL 2016
    (https://aclweb.org/anthology/P/P16/P16-2058.pdf)

• Hybrid Word-Character Models
  – Luong and Manning, ACL 2016
    (https://www.aclweb.org/anthology/P/P16/P16-1100.pdf)

• Other segmentations
Byte-Pair Encoding (BPE)


Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016.
Neural Machine Translation of Rare Words with Subword
Units. In Proceedings of the 54th Annual Meeting of the
Association for Computational Linguistics (ACL 2016), Berlin,
Germany.
(https://www.aclweb.org/anthology/P/P16/P16-1162.pdf)

Code: https://github.com/rsennrich/subword-nmt

Example:
Thank You

Contact:

Rebecca Knowles
rknowles@jhu.edu
www.cs.jhu.edu/~rknowles/

This work was supported, in part, by a National Science Foundation Graduate Research Fellowship under Grant No. DGE-1232825. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect those of the sponsors.