The Sockeye Neural MT Toolkit at AMTA 2018

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, Matt Post

github.com/awslabs/sockeye
Why Sockeye?

Sockeye is:
• A production-ready framework for training state-of-the-art models
• A flexible experimentation platform for researchers

Motivation: rapid evolution of Neural MT—different toolkits with different features
• No single toolkit with everything we need at Amazon
• Nothing mature for MXNet, our framework of choice

Decision: build such a toolkit—Sockeye
• Highly scalable (multiple GPUs, large data)
• Free and open source software (Apache 2.0)

Named after the *Sockeye* salmon found in the Northern Pacific Ocean
(Favorite fish around Seattle, WA)
Quick Start

A translation system in 3 slides
Sequence-to-Sequence Modeling

Language model conditioned on source sentence $x = x_1, ..., x_m$:

$$p(y|x) = \prod_{t=1}^{n} p(y_t|y_{1:t-1}, x)$$

**Encode** source sentence

**Decode** target sentence

**Attention** connects states across steps

Many instantiations:
- Recurrent
- Convolutional
- Self-attentional
Data Pre-Processing

Given raw parallel text:

The shares closed almost unchanged at 187.35 dollars.
The question comes alone: Collserola? Park or mountain?

Step 1 – Tokenize:

The shares closed almost unchanged at 187.35 dollars.
The question comes alone: Collserola? Park or mountain?

Step 2 – Sub-word encode:

The shares closed almost unchanged at 187.35 dollars.
The question comes alone: Collserola? Park or mountain?

Ready for training!
Running Sockeye

**Install Sockeye:**
pip install sockeye

**Train with default settings:**
python -m sockeye.train \ 
  --source train-corpus.de \ 
  --target train-corpus.en \ 
  --validation-source dev-corpus.de \ 
  --validation-target dev-corpus.en \ 
  --output model.de-en

**Decode with default settings:**
python -m sockeye.translate \ 
  --models model.de-en

**Customization?**
Architectures & Features

Customizing translation systems
Sockeye supports 3 prominent architectures:
- Mix and match with `--encoder` and `--decoder` options

- **Attentional Recurrent** [Bahdanau et al., 2014, Luong et al., 2015]
- **Fully Convolutional** [Gehring et al., 2017]
- **Self-Attentional Transformer** [Vaswani et al., 2017]
## Attention Types

Sockeye supports a range of attention models (currently limited to RNN encoders/decoders)

<table>
<thead>
<tr>
<th>Name</th>
<th>score(s, h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP [Bahdanau et al, 2014]</td>
<td>$v_a^T \tanh(W_u s + W_v h)$</td>
</tr>
<tr>
<td>Dot [Luong et al. 2015]</td>
<td>$s^T h$</td>
</tr>
<tr>
<td>Location [Luong et al. 2015]</td>
<td>$v_a s$</td>
</tr>
<tr>
<td>Bilinear [Luong et al. 2015]</td>
<td>$s^T W h$</td>
</tr>
<tr>
<td>Coverage [Tu et al. 2015]</td>
<td>$v_a^T \tanh(W_u s + W_v h + W_c C)$</td>
</tr>
<tr>
<td>Multi-head [Vaswani et al., 2017]</td>
<td>$\text{softmax} \left( \frac{s W_u^Q (h W_u^K)^T}{\sqrt{d_u}} \right) h W_u^V$</td>
</tr>
</tbody>
</table>
Training

Recommended model training recipe:

• Adam optimizer with learning rate scheduler
• Learning rate reduces when dev perplexity plateaus
• Decay resets model and optimizer parameters to best point
• Early stopping on extended dev perplexity plateau
• Average model parameters from best checkpoints
Training

Recommended model training recipe:

• Adam optimizer with learning rate scheduler
  - Optimizers: SGD, Nadam [Dozat, 2015], Eve [Koushik and Hayashi, 2016], etc.
  - Multi-GPU parallelization with sentence or word-based batching
  - Training resumption, sharding + serialized preprocessed data
  - Factored input [Sennrich and Haddow, 2016]

• Learning rate reduces when dev perplexity plateaus
  - Fixed-step, inverse-square-root [Vaswani et al., 2017] and more

• Decay resets model and optimizer parameters to best point
  - Alternatively restart optimizer (momentum) from zero [Denkowski and Neubig, 2017]

• Early stopping on extended dev perplexity plateau
  - Alternatively track BLEU, chrF [Popović, 2015], etc., or train for set number of updates

• Average model parameters from best checkpoints
Monitoring

Monitor training with standalone TensorBoard:
• BLEU, chrF, and perplexity curves for different model configurations
• Easily add and track new metrics
Decoding

Primary decoding features:

• Length-normalized beam search
  - Parametrized length penalty [Wu et al., 2016]
  - Target vocabulary selection [Devlin, 2017]

• Efficient GPU and CPU support
  - Length-based batch decoding

• Ensemble multiple models
  - Including different architectures

• Visualize system output
  - Attention matrices (alignments)
Decoding

Visualize beam search history

Adding new features?
Development

Adding your code to Sockeye
Starting with **MXNet**

Fast and scalable deep learning framework
* Native support for parallelization of training
* Near linear speedup with multiple GPUs

Flexible programing model
* Imperative API (NumPy on GPUs)
* Symbolic API (computation graphs)

Bindings for various languages (Python, C++, Scala, R, Julia, Perl)

Officially supported by Amazon/AWS
* Quick start with Amazon Deep Learning AMI
MXNet Programming Models

**Imperative**

- Like NumPy, but GPU backend

```python
from mxnet.ndarray import *
x = zeros((64, 12))
weights = zeros((128, 12))
x = FullyConnected(
    x, weights, num_hidden=128)
pred = SoftmaxActivation(x)
pred = pred.asnumpy()
```

**Symbolic**

- Optimized computation graph, auto-diff

```python
from mxnet.symbol import *
y = Variable('y')
x = Variable('x')
weights = Variable('w')
x = FullyConnected(
    x, weights, num_hidden=128)
pred = SoftmaxOutput(x, y)
model = Module(pred)
model.fit(...)
model.forward_backward(data)
```
Implementation

Training - Symbolic:
- Unroll models through time to maximum sequence length
- Organize data into buckets of similar length
- One symbolic graph per bucket with shared memory and parameters

Inference - Symbolic and Imperative:
- Symbolic: encode source sequence
- Imperative: iteratively generate target
- Beam search decoder maintains & expands k-best hypotheses at each step until <EOS>
Developing Sockeye

Official Amazon software on GitHub
Developing Sockeye

Developer guidelines for reliable, understandable code:
• Python3 with type annotations and Sphinx-style doc strings
• Comprehensive unit and system tests
• Peer review and code documentation
Developing Sockeye

Public code review process—community feedback welcome!

Source factors #275

mjpost commented on Jan 18 • edited by fliher

Added source factors, as described in.

Linguistic Input Features Improve Neural Machine Translation.
Rico Sennrich & Barry Haddow

Source factors are enabled by passing --source-factors file1 [file2 ...] (-sf), where file1, etc. are token-parallel to the source (-s).
The Sockeye Neural MT Toolkit at AMTA 2018

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, Matt Post

github.com/awslabs/sockeye
## WMT17 News Translation Task

<table>
<thead>
<tr>
<th>System</th>
<th>Architecture</th>
<th>EN→DE</th>
<th>LV→EN</th>
</tr>
</thead>
<tbody>
<tr>
<td>FairSeq</td>
<td>CNN</td>
<td>23.37</td>
<td>15.38</td>
</tr>
<tr>
<td>Marian</td>
<td>RNN</td>
<td>25.93</td>
<td>16.19</td>
</tr>
<tr>
<td></td>
<td>Transformer</td>
<td><strong>27.41</strong></td>
<td><strong>17.58</strong></td>
</tr>
<tr>
<td>Nematus</td>
<td>RNN</td>
<td>23.78</td>
<td>14.70</td>
</tr>
<tr>
<td>Neural Monkey</td>
<td>RNN</td>
<td>13.73</td>
<td>10.54</td>
</tr>
<tr>
<td>OpenNMT</td>
<td>RNN</td>
<td>22.69</td>
<td>13.85</td>
</tr>
<tr>
<td>OpenNMT-py</td>
<td>RNN</td>
<td>21.95</td>
<td>13.55</td>
</tr>
<tr>
<td>Tensor2Tensor</td>
<td>Transformer</td>
<td>26.34</td>
<td>17.67</td>
</tr>
<tr>
<td>Sockeye</td>
<td>CNN</td>
<td>24.59</td>
<td>15.82</td>
</tr>
<tr>
<td></td>
<td>RNN</td>
<td>25.55</td>
<td>15.92</td>
</tr>
<tr>
<td></td>
<td>Transformer</td>
<td><strong>27.50</strong></td>
<td><strong>18.06</strong></td>
</tr>
</tbody>
</table>

WMT BLEU (cased)