mt@amazon

# What's New in Sockeye?

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar,
Artem Sokolov, Ann Clifton, Matt Post, Daniel Vidal, Kellen Sunderland

github.com/awslabs/sockeye

# Sockeye NMT Toolkit

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

- 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

# Quick Start

A translation system in 3 slides

# Sequence-to-Sequence Modeling

Language model **conditioned on source sentence** $\boldsymbol{x} = x_1, \dots, x_m$:

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

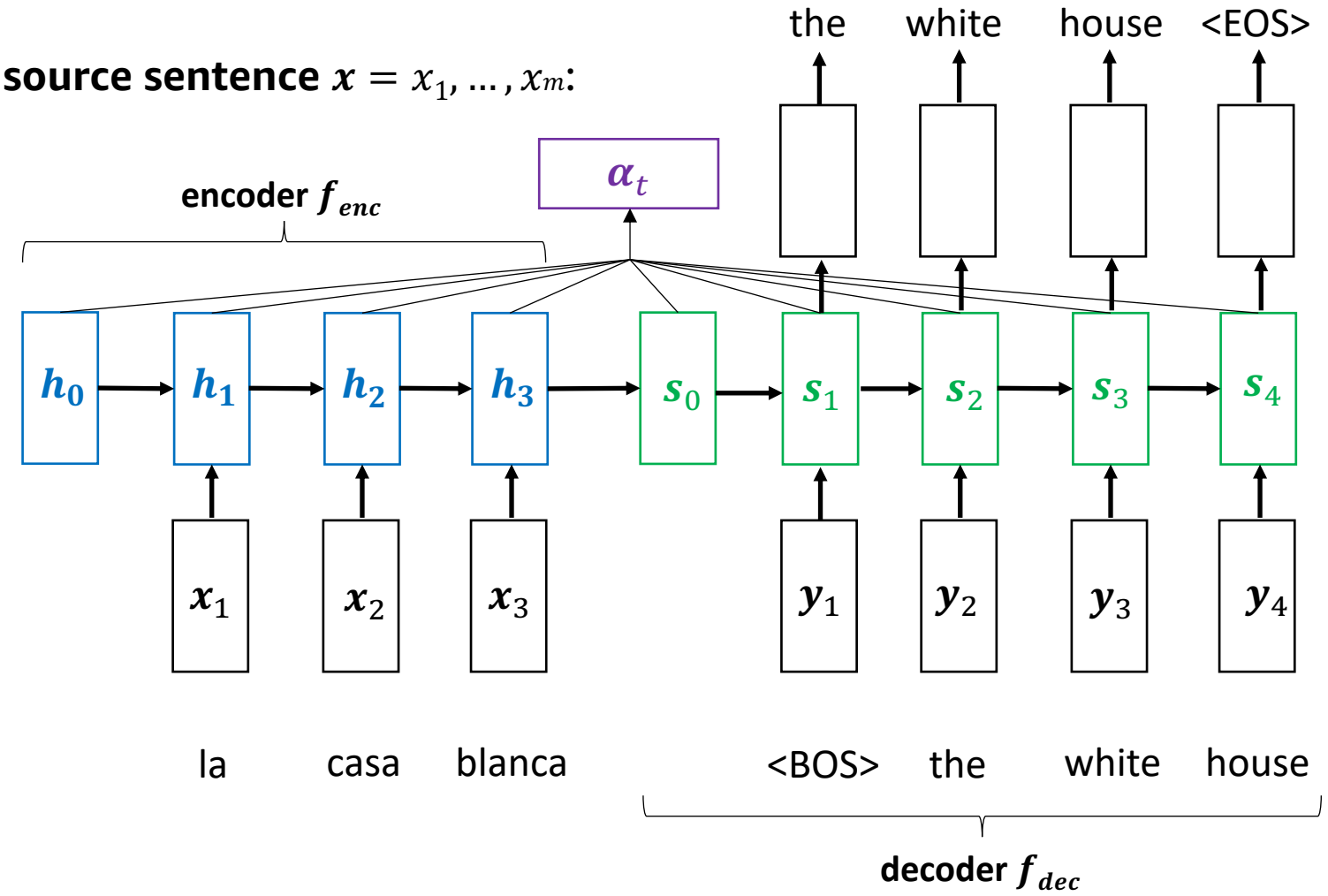
Encode source sentence

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 share@@ s closed a@@ lmost un@@ chang@@ ed at 18@@ 7@@ .@@ 35 dollar@@ s .
The question comes alone : Co@@ ll@@ s@@ er@@ ola ? 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?**

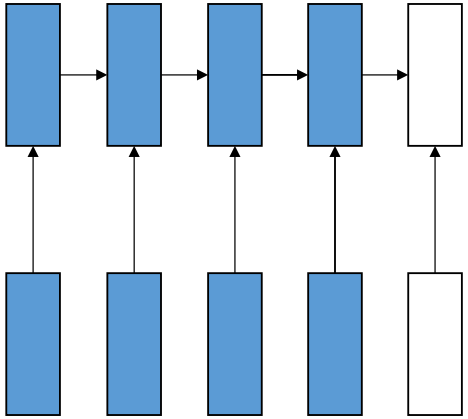
# Architectures & Features

Customizing translation systems

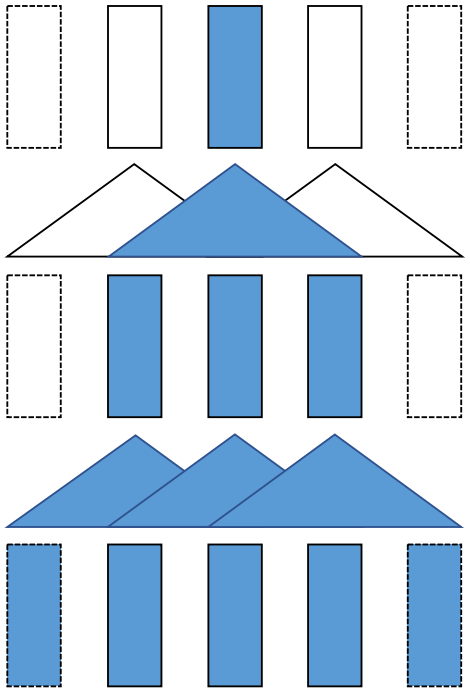
# Base Architectures

Sockeye supports 3 prominent architectures:

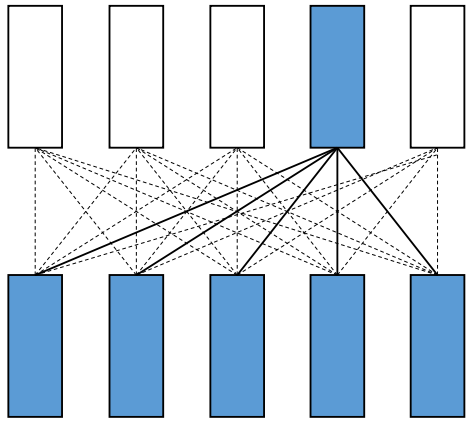
- Mix and match with **`--encoder`** and **`--decoder`** options



Attentional Recurrent
[Bahdanau et al., 2014, Luong et al., 2015]



Fully Convolutional
[Gehring et al., 2017]



Self-Attentional Transformer
[Vaswani et al., 2017]

# 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

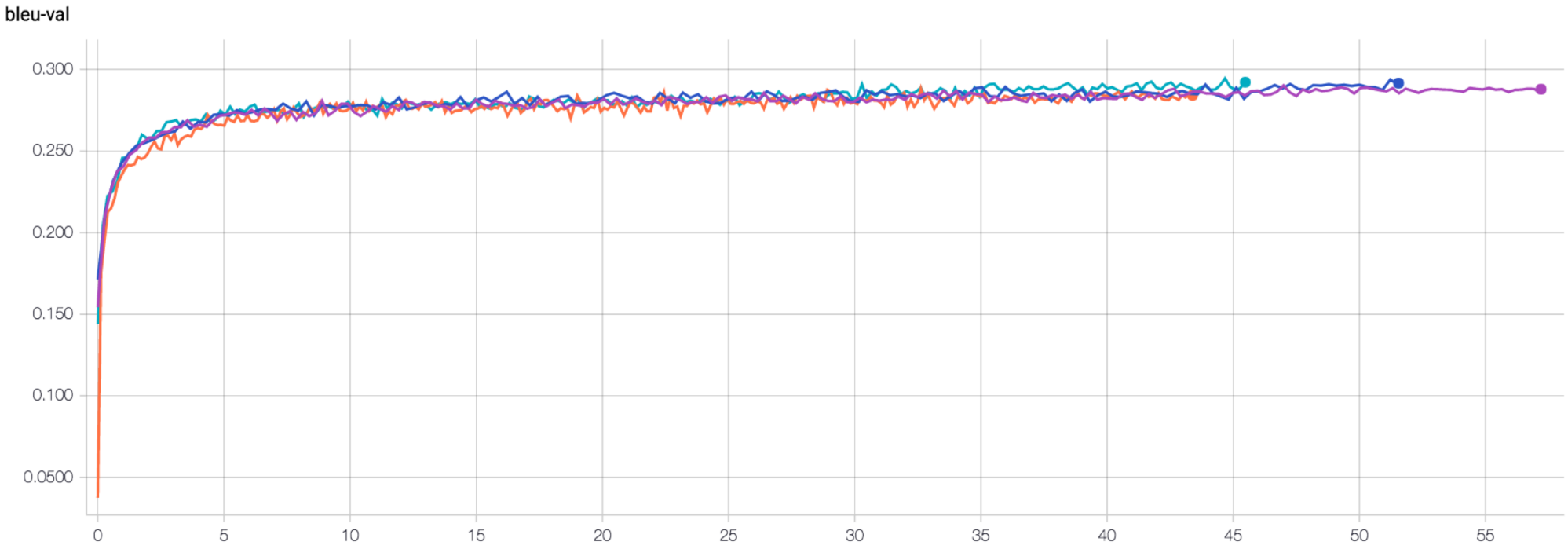
Recommended model training recipe:

- 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
  - or restart optimizer (momentum) from zero [Denkowski and Neubig, 2017]
- Early stopping on extended dev perplexity plateau
  - or 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:

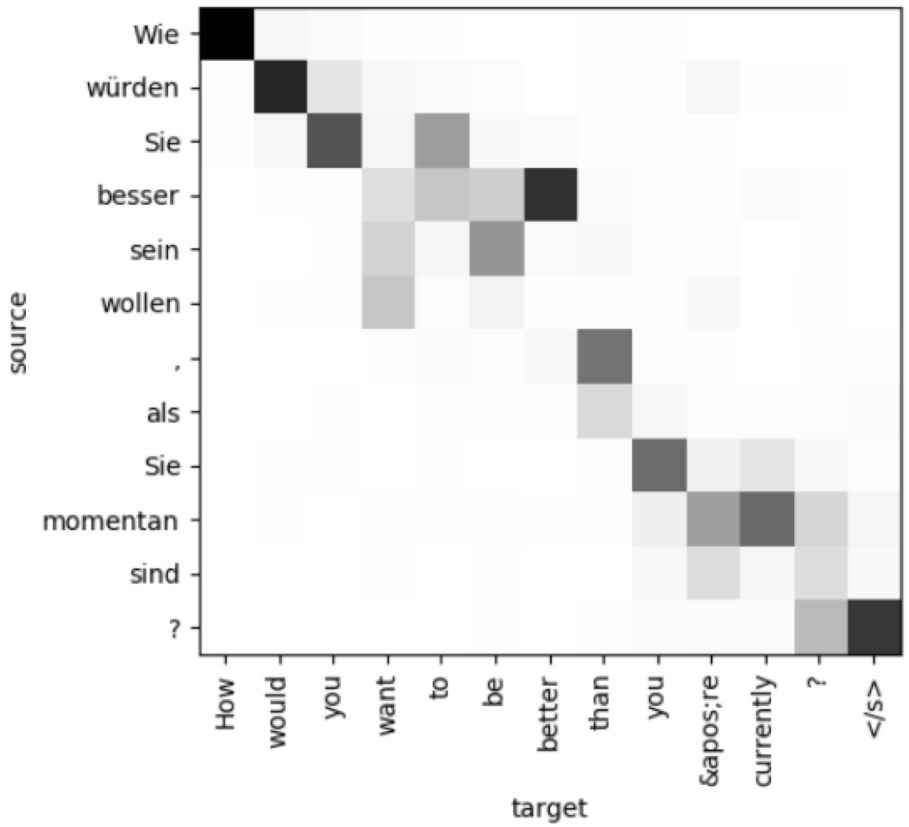
- BLEU, chrF, and perplexity curves for different model configurations
- Easily add and track new metrics



# Decoding

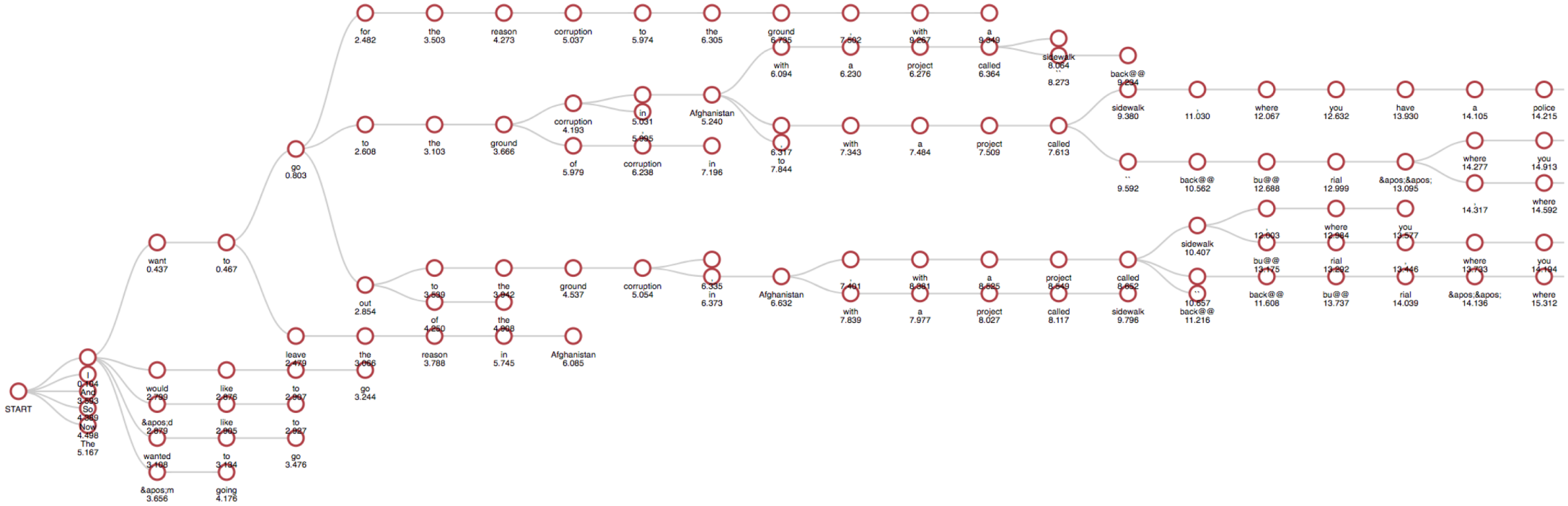
Primary decoding features:

- 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



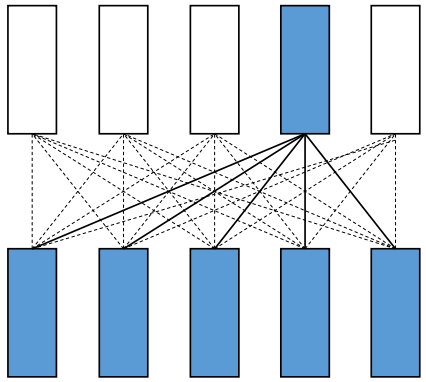
# Recent Work

Newest features and updates

# Performance Improvements

Transformer updates

- Default settings updated to strongest model from arXiv paper
- Refactoring to optimize computation graphs for attention, decoding
- Improved batching for beam search



[Preview] Update to MXNet 1.2

- New LayerNormalization operator saves GPU memory, ~20% speedup
- General speed optimizations for GPU and CPU, many from Amazon's Core Machine Learning, AWS teams

# New Features

"Fast Lexically Constrained Decoding with Dynamic Beam Allocation for Neural Machine Translation" [Post and Vilar, 2018]

```
{ "text": "Einer soll ein hoch@@ rangi@@ ges Mitglied aus Berlin gewesen sein .",
  "constraints": ["is said to", "powerful"] }
```

"Learning Hidden Unit Contribution for Adapting Neural Machine Translation Models" [Vilar 2018]

- Adapt models with small number of additional parameters, adaptation data
- Load/freeze different model parameters during training

[Coming Soon] FP16 support for training and inference

- 2X-3X reported speedup for deep learning applications on NVIDIA GPUs

# Building NMT Systems

```
wget http://www.statmt.org/wmt13/training-parallel-europarl-v7.tgz
wget http://www.statmt.org/wmt13/training-parallel-commoncrawl.tgz
wget http://data.statmt.org/wmt17/translation-task/training-parallel-nc-v12.tgz
wget http://data.statmt.org/wmt17/translation-task/rapid2016.tgz
wget http://data.statmt.org/wmt17/translation-task/dev.tgz
wget http://data.statmt.org/wmt17/translation-task/test.tgz

tar xf training-parallel-europarl-v7.tgz
tar xf training-parallel-commoncrawl.tgz
tar xf training-parallel-nc-v12.tgz
tar
tar
tar

cat
commentary-v12.de-en.de rapid2016.de-en.de >train.de
cat training/europarl-v7.de-en.en commoncrawl.de-en.en training/news-
commentary-v12.de-en.en rapid2016.de-en.en >train.en

stripsgml <dev/newstest2016-deen-src.de.sgm > dev.de
stripsgml <dev/newstest2016-deen-ref.en.sgm > dev.en
stripsgml <test/newstest2017-deen-src.de.sgm > test.de
stripsgml <test/newstest2017-deen-ref.en.sgm > test.en

./tokenizer.perl -l de <train.de >train.de.tok
./tokenizer.perl -l de <dev.de >dev.de.tok
./tokenizer.perl -l de <test.de >test.de.tok
./tokenizer.perl -l en <train.en >train.en.tok
./tokenizer.perl -l en <dev.en >dev.en.tok
./tokenizer.perl -l en <test.en >test.en.tok

cat train.de.tok train.en.tok |./learn_bpe.py -s 3200 >codes

./apply_bpe.py -c codes train.de.tok >train.de.tok.bpe
./apply_bpe.py -c codes train.de.tok >train.de.tok.bpe
```

```
./apply_bpe.py -c codes train.de.tok >train.de.tok.bpe
./apply_bpe.py -c codes train.de.tok >train.de.tok.bpe
./apply_bpe.py -c codes train.de.tok >train.de.tok.bpe
./apply_bpe.py -c codes train.de.tok >train.de.tok.bpe

python3 -m sockeye.train -s train.de.bpe -t train.en.bpe -vs dev.de.bpe -vt
dev.en.bpe -o model --seed=1 --batch-type=word --batch-size=4092 --checkpoint-
frequency=2000 --device-ids=-4 --embed-dropout=0:0 --encoder=transformer --
decoder=transformer --num-layers=6:6 --transformer-model-size=512 --
transformer-attention-heads=8 --transformer-feed-forward-num-hidden=2048 --
-
init-xavier-factor-type=avg --embed-weight-init=default --num-embed=512:512 --
optimizer=adam --optimized-metric=perplexity --gradient-clipping-threshold=-1 -
-gradient-clipping-type=abs --initial-learning-rate=0.0002 --learning-rate-
reduce-num-not-improved=8 --learning-rate-reduce-factor=0.9 --learning-rate-
scheduler-type=plateau-reduce --learning-rate-warmup=0 --learning-rate-decay-
optimizer-states-reset=best --learning-rate-decay-param-reset --max-num-
checkpoint-not-improved=32 --min-num-epochs=1 --decode-and-evaluate=500 --keep-
last-params=60

python3 -m sockeye.average -n 8 --output=model/params.average.best --
strategy=best model

cat test.en |./apply_bpe.py -c codes |python3 -m sockeye.translate --
models=model --beam-size=5 --batch-size=32 --chunk-size=1000 --length-penalty-
alpha=0.1 --length-penalty-beta=0.0 --max-output-length-num-stds=2 --bucket-
width=10 |sed -u -r 's/@@( |\$)//g' >test.out

python3 -m sockeye.evaluate --hypotheses=test.out --references=test.de
>test.scores
```

```
sockeye-autopilot --task wmt17_de_en --model transformer
```

# Autopilot

```
sockeye-autopilot --task wmt17_de_en --model transformer
```

- Download and preprocesses data with Moses tokenizer, subword-nmt byte-pair encoding
- Train and evaluate translation model
- Log commands, intermediate files available as plain text

```
sockeye-autopilot --custom-task my_task --custom-train train.src train.trg
--custom-dev dev.src dev.trg --custom-test test.src test.trg --model transformer
```

- Automatic training and evaluation on user data (raw, tokenized, or BPE)

```
sockeye-autopilot --task wmt17_de_en --model none
```

- Download and preprocess standard data set for other use

# Development

Adding your code to Sockeye

# Starting with mxnet

amazon

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)

Officially supported by Amazon/AWS

- Quick start with Amazon Deep Learning AMI

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

# MXNet Programming Models

## Imperative

- Like NumPy on GPUs

```
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 graphs

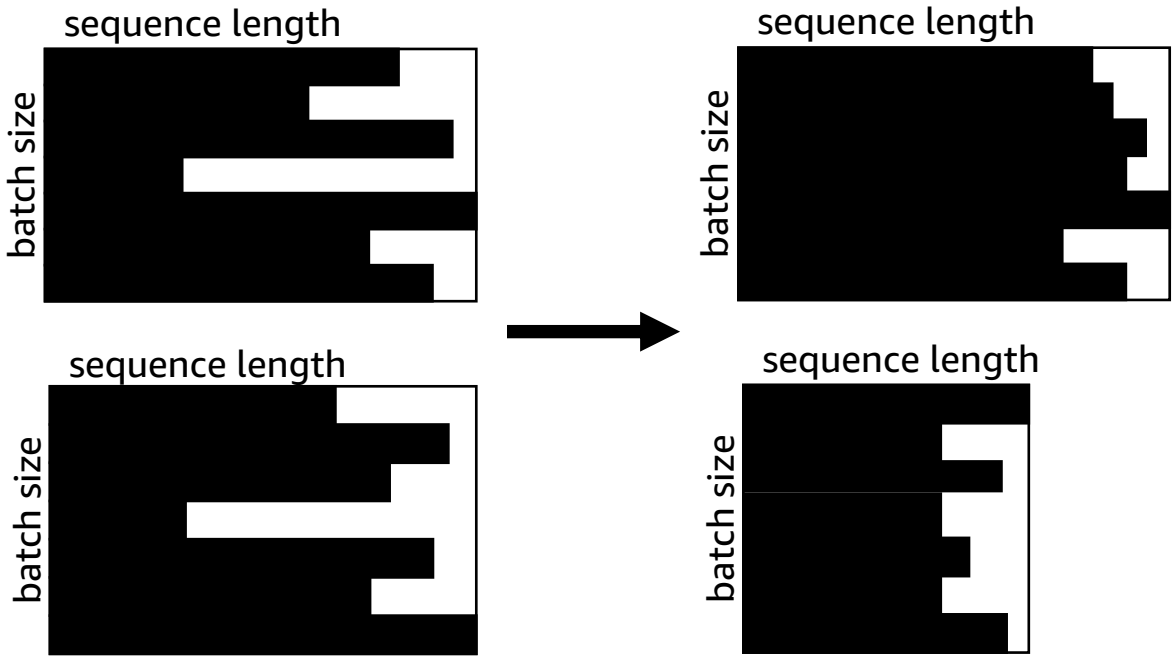
```
from mxnet.symbol import *

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

awslabs / sockeye

Unwatch 43 | Star 508 | Fork 140

Code | Issues 4 | Pull requests 9 | Projects 0 | Insights | Settings

Sequence-to-sequence framework with a focus on Neural Machine Translation based on Apache MXNet

Edit

deep-learning deep-neural-networks mxnet machine-learning machine-translation neural-machine-translation encoder-decoder attention-mechanism sequence-to-sequence sequence-to-sequence-models sockeye attention-is-all-you-need attention-alignment-visualization attention-model seq2seq convolutional-neural-networks translation Manage topics

307 commits | 11 branches | 29 releases | 20 contributors | Apache-2.0

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

fhieber Simplified computation of smoothed cross entropy loss in CrossEntropy... ...

Latest commit 74804ce a day ago

# Developing Sockeye

Developer guidelines for reliable, understandable code:

- Python3 with type annotations, Sphinx-style doc strings
- Peer review and code documentation

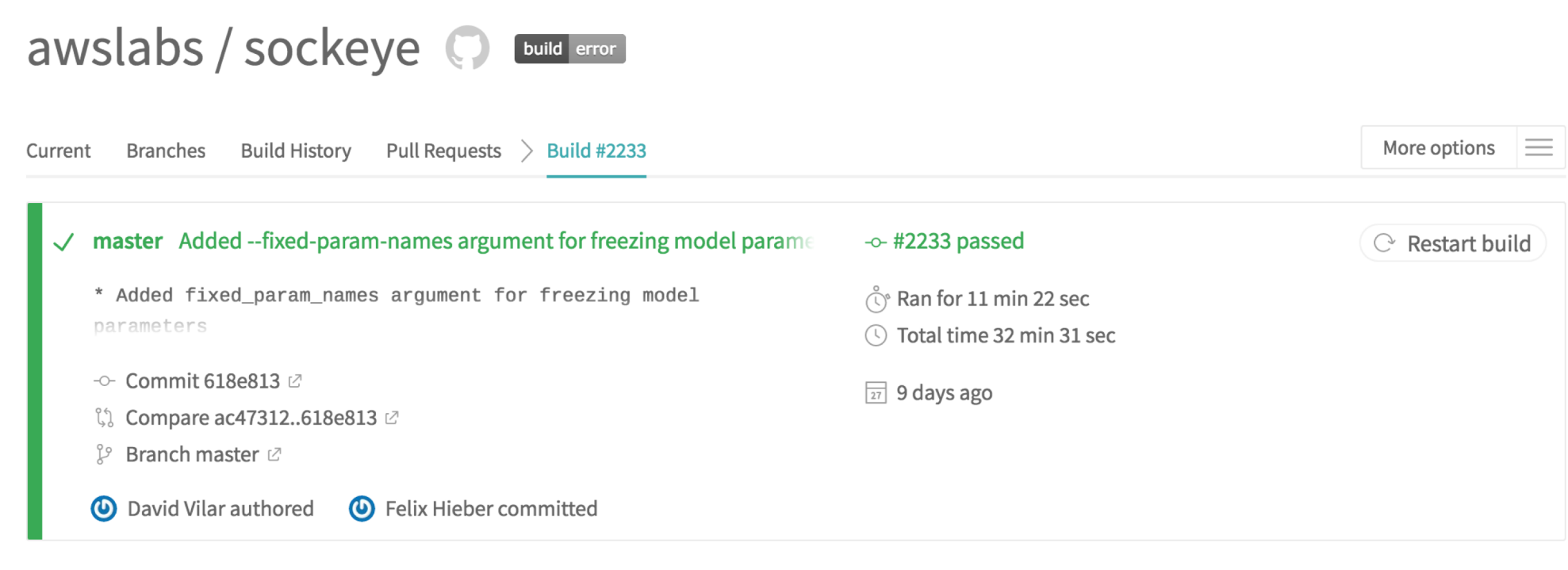
```python
def split_heads(x: mx.sym.Symbol, depth_per_head: int, heads: int) -> mx.sym.Symbol:
    """
    Returns a symbol with head dimension folded into batch and depth divided by the number of heads.

    :param x: Symbol of shape (batch, length, depth).
    :param depth_per_head: Depth per head.
    :param heads: Number of heads.
    :return: Symbol of shape (batch * heads, length, depth_per_heads).
    """
    # (batch, length, heads, depth_per_head)
    x = mx.sym.reshape(data=x, shape=(0, -1, heads, depth_per_head))
    # (batch, heads, length, depth/heads)
    x = mx.sym.transpose(data=x, axes=(0, 2, 1, 3))
    # (batch * heads, length, depth/heads)
    return mx.sym.reshape(data=x, shape=(-3, -1, depth_per_head))
```

# Developing Sockeye

Comprehensive automated testing

- Unit, integration, and system tests for all features
- Tests run automatically for each code change



# Developing Sockeye

Public code review process—community feedback welcome!

## Source factors #275

Merged **fhieber** merged 77 commits into `master` from `source_factors` 18 days ago

Conversation 230 | Commits 77 | Files changed 29

**mjpost** commented on Jan 18 • edited by fhieber — Collaborator

Added source factors, as described in.

```
Linguistic Input Features Improve Neural Machine Translation.
Rico Sennrich & Barry Haddow
In Proceedings of the First Conference on Machine Translation. Berlin, Germany, pp. 83-91.
```

Source factors are enabled by passing `--source-factors file1 [file2 ...]` ( `-sf` ), where file1, etc. are token-parallel to the source ( `-s` ).

Data cleanup (towards source_factors PR) (#293) Verified b74fbd0

removed comment fd86a09

Bumped Major Version to 1.17.0 6b095ea

**fhieber** commented 27 days ago • edited — Member

As far as I can tell, I addressed all comments. **@davvil @tdomhan** I would merge once both of you approve.

**tdomhan** requested changes 27 days ago — View changes

sockeye/inference.py — Show outdated

sockeye/inference.py — Show outdated

sockeye/inference.py — Show outdated

sockeye/inference.py — Show outdated

**davvil** approved these changes 27 days ago — View changes

**fhieber** added some commits 27 days ago

Do not throw exceptions while translating (#294) Verified 1a48ebe

docstring & clarity 18aa317

**fhieber** referenced this pull request 25 days ago

**Remove bias parameters in Transformer attention layers as they bring no benefit** #296 — Merged

7 of 7 tasks complete

Remove bias parameters in Transformer attention layers as they bring … Verified 7d958c1

**tdomhan** reviewed 22 days ago — View changes

sockeye/arguments.py

```
+    decode_params.add_argument('--json-input',
+                               action='store_true',
+                               default=False,
+                               help="If given, the CLI
```

**tdomhan** 22 days ago — Member

as this is the only place where a user gets information about the input format without looking at the source code we should fully describe the format here. Namely, which key/value pairs one expects and maybe a tiny example with factors..

**fhieber** 22 days ago — Member

done

mt@amazon

# What's New in Sockeye?

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar,
Artem Sokolov, Ann Clifton, Matt Post, Daniel Vidal, Kellen Sunderland

github.com/awslabs/sockeye