1 year developing and maintaining OpenNMT

Guillaume Klein

OpenNMT Workshop Paris, March 2018

1 year of OpenNMT…

… short for machine translation

SYSTRAN back to 1968

… long for open source

How to keep people interested and invested?

… very long for deep learning

- New frameworks
- New hardware
- New models

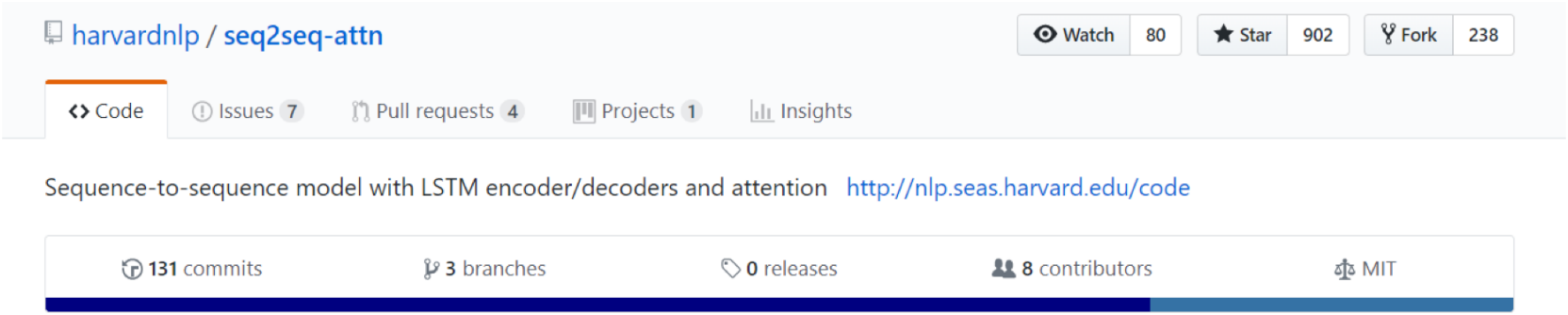
1 year developing and maintaining OpenNMT

Or:

Why is OpenNMT still relevant 1 year later?

Chapter 1: Building on strong basis

The seq2seq-attn legacy



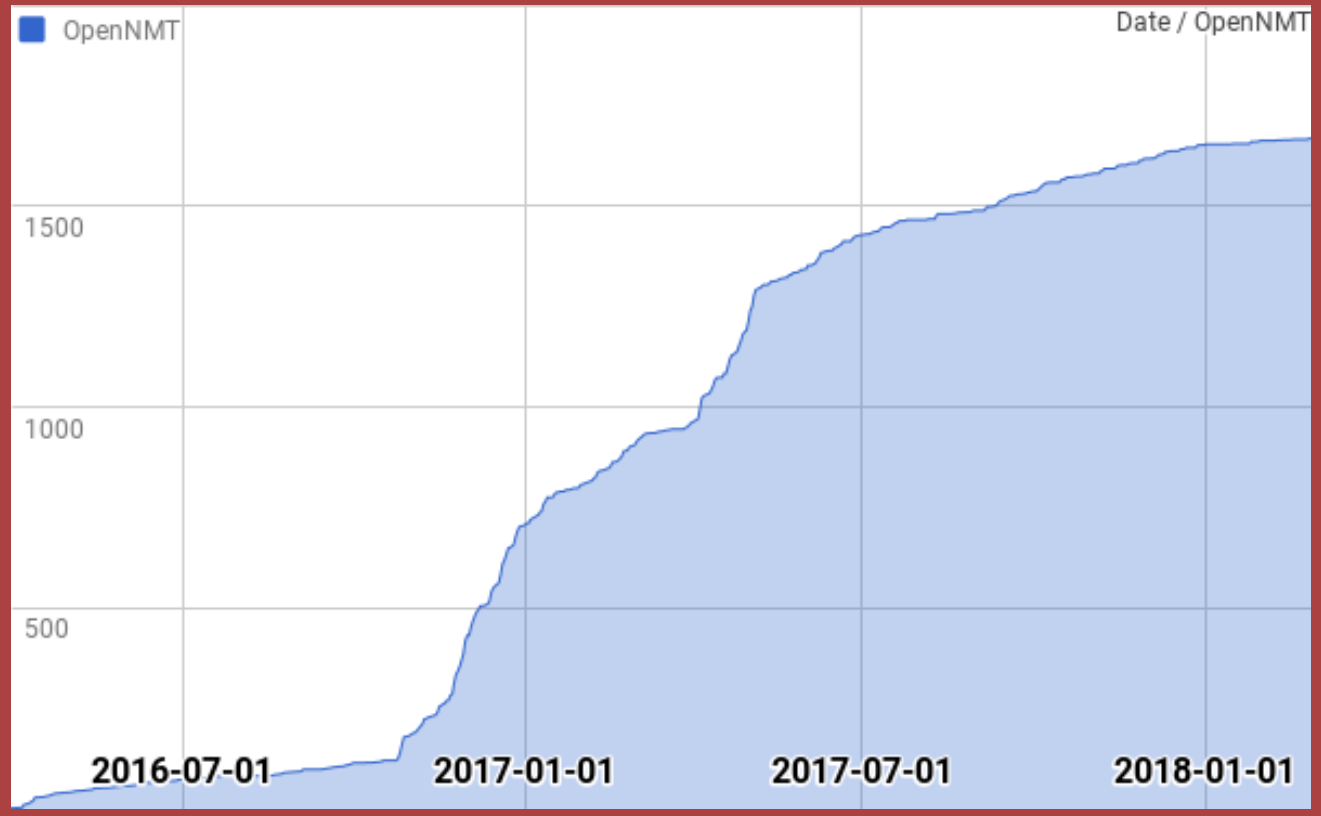
Efficient

State-of-the-art

Easy to use

➔ OpenNMT: **a fork** not an inspiration

Chapter 2: Growing fast



Evolution of the number of commits in OpenNMT

Lots of new features…

Tokenization tools

New encoder types

Multi GPU training

Input vectors

Translation servers

Validation metrics

Dynamic dataset

…

v0.1.0 (12/16)

v0.2.0 (01/17)

v0.3.0 (01/17)

v0.4.0 (02/17)

v0.5.0 (03/17)

v0.6.0 (04/17)

v0.7.0 (05/17)

v0.8.0 (06/17)

v0.9.0 (11/17)

… lots of new challenges

Growing complexity

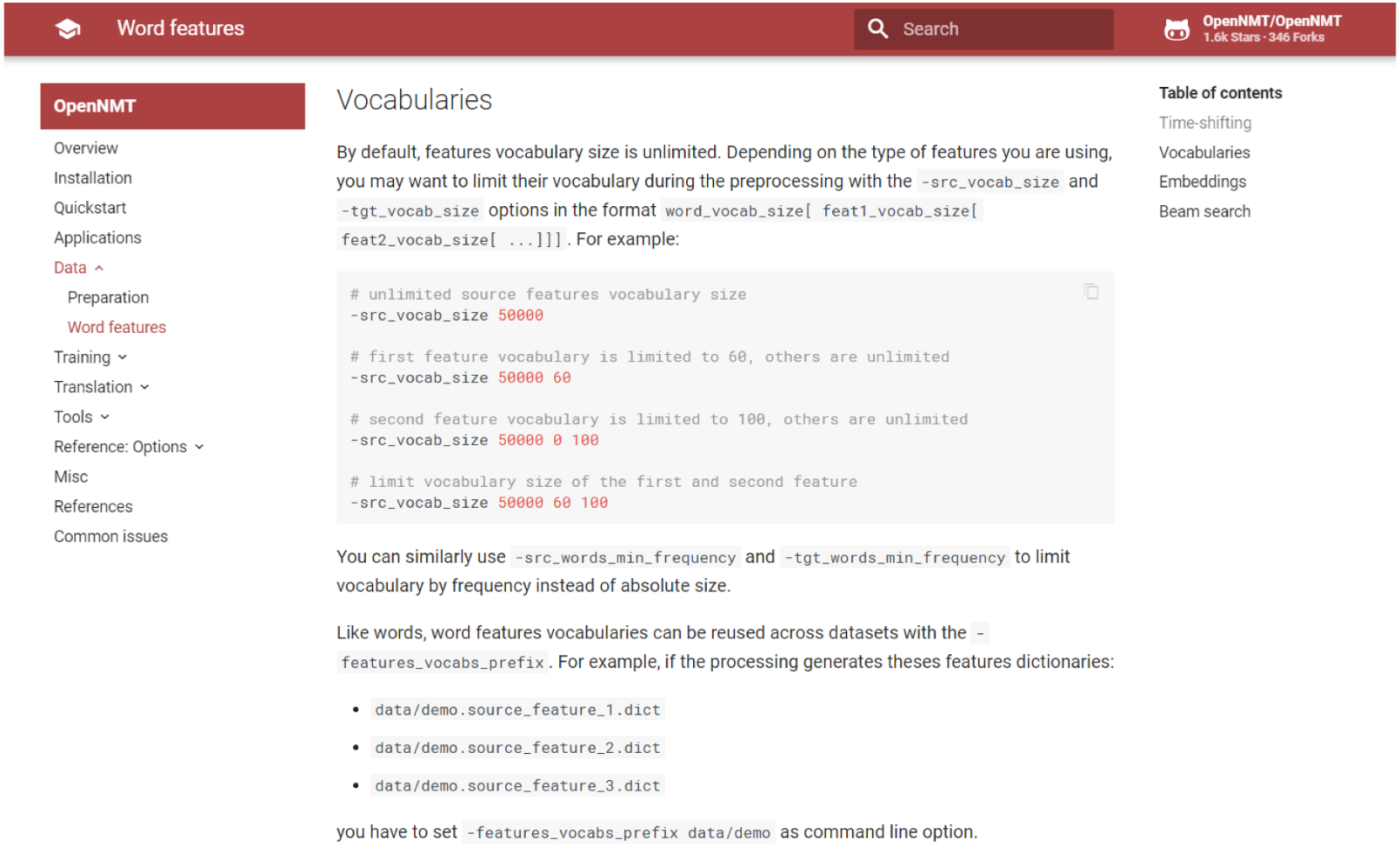
Versioning semantic

- **Rule 1:** Never break model compatibility
- **Rule 2:** Do not break translation APIs unless required
- **Rule 3:** Avoid breaking options name
- **Rule 4:** No stability guarantes for the code

Difficult balance between:

- **Research interest:** *custom models, advanced training flow, extensibility*
- **Industry interest:** *robustness, stability, integration*
- **User interest:** *ease of use, low hardware requirements*

Documentation



Word features

Search

OpenNMT/OpenNMT
1.6k Stars · 346 Forks

OpenNMT

- Overview
- Installation
- Quickstart
- Applications
- Data
  - Preparation
  - Word features
- Training
- Translation
- Tools
- Reference: Options
- Misc
- References
- Common issues

Vocabularies

By default, features vocabulary size is unlimited. Depending on the type of features you are using, you may want to limit their vocabulary during the preprocessing with the `-src_vocab_size` and `-tgt_vocab_size` options in the format `word_vocab_size[ feat1_vocab_size[ feat2_vocab_size[ ...]]]`. For example:

```
# unlimited source features vocabulary size
-src_vocab_size 50000

# first feature vocabulary is limited to 60, others are unlimited
-src_vocab_size 50000 60

# second feature vocabulary is limited to 100, others are unlimited
-src_vocab_size 50000 0 100

# limit vocabulary size of the first and second feature
-src_vocab_size 50000 60 100
```

You can similarly use `-src_words_min_frequency` and `-tgt_words_min_frequency` to limit vocabulary by frequency instead of absolute size.

Like words, word features vocabularies can be reused across datasets with the `-features_vocabs_prefix`. For example, if the processing generates theses features dictionaries:

- `data/demo.source_feature_1.dict`
- `data/demo.source_feature_2.dict`
- `data/demo.source_feature_3.dict`

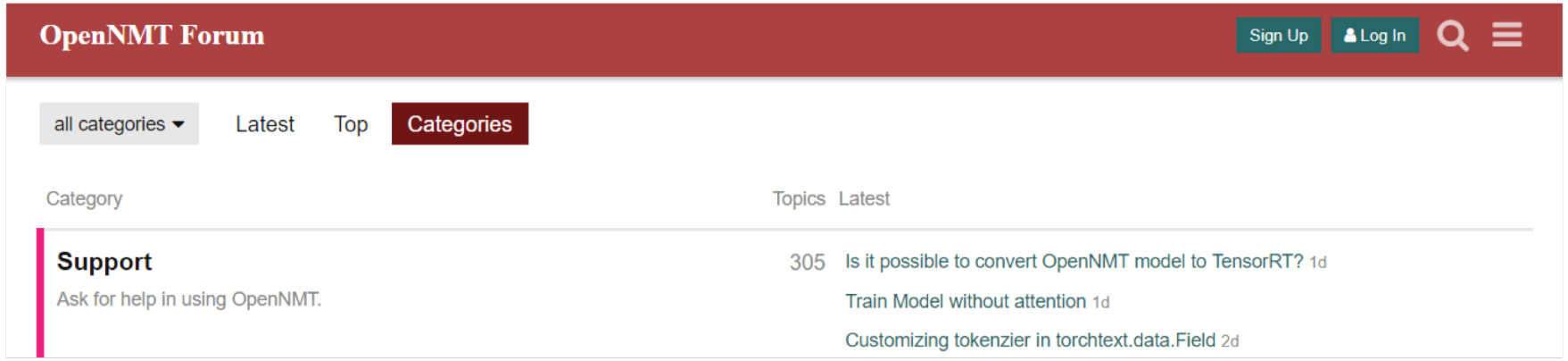
you have to set `-features_vocabs_prefix data/demo` as command line option.

Table of contents

- Time-shifting
- Vocabularies
- Embeddings
- Beam search

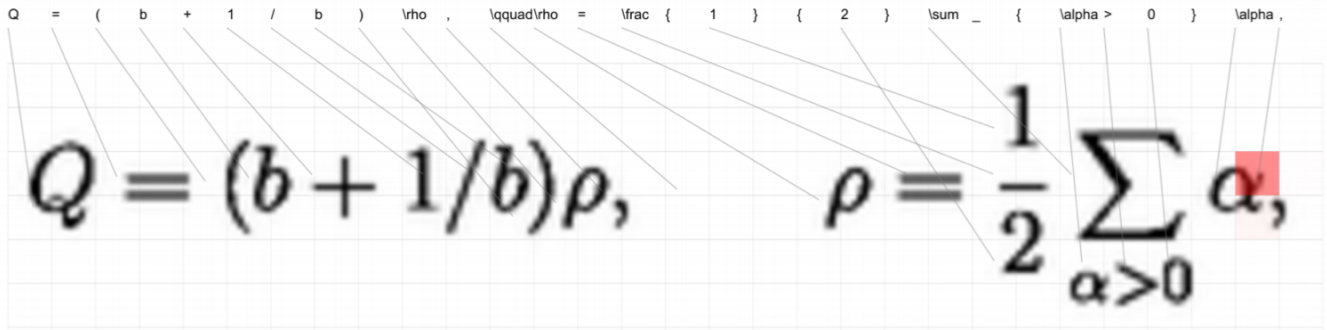
User support

Goal: answering and fixing issues fast.



Chapter 3: Extending the core project

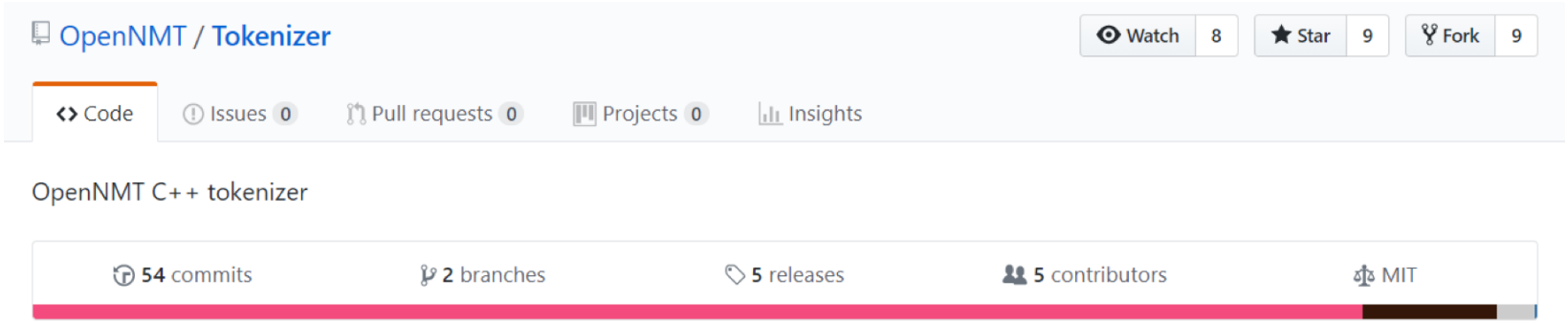
OpenNMT/Im2Text



"A deep learning-based approach to learning the image-to-text conversion, built on top of the OpenNMT system."

—Yuntian Deng

OpenNMT/Tokenizer



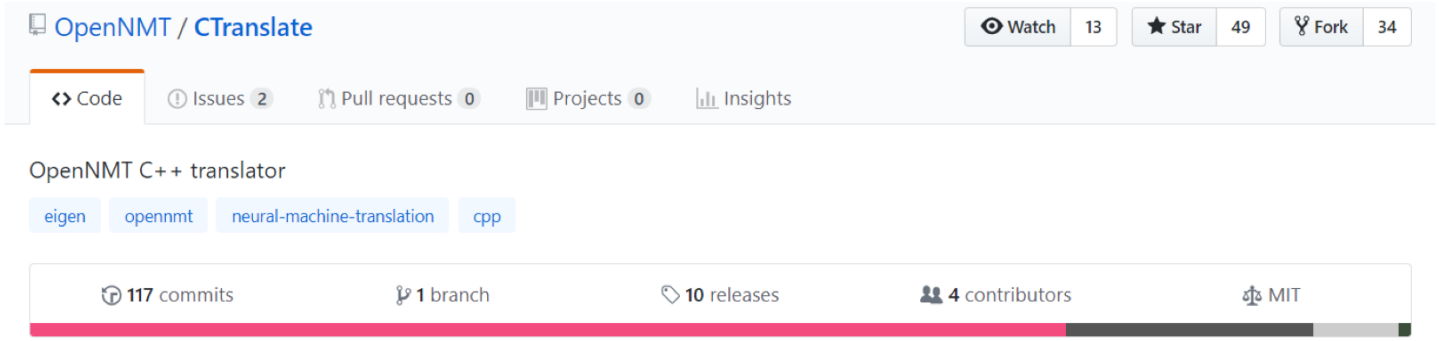

```python
import pyonmttok

tokenizer = pyonmt.Tokenizer(
    mode: str,
    bpe_model_path="",
    joiner="▪",
    joiner_annotate=False,
    joiner_new=False,
    case_feature=False,
    no_substitution=False,
    segment_case=False,
    segment_numbers=False,
    segment_alphabet_change=False,
    segment_alphabet=[])

tokens, features = tokenizer.tokenize(test: str)

text = tokenizer.detokenize(tokens, features)
text = tokenizer.detokenize(tokens)  # will fail if case_feature is set.
```

OpenNMT/CTranslate



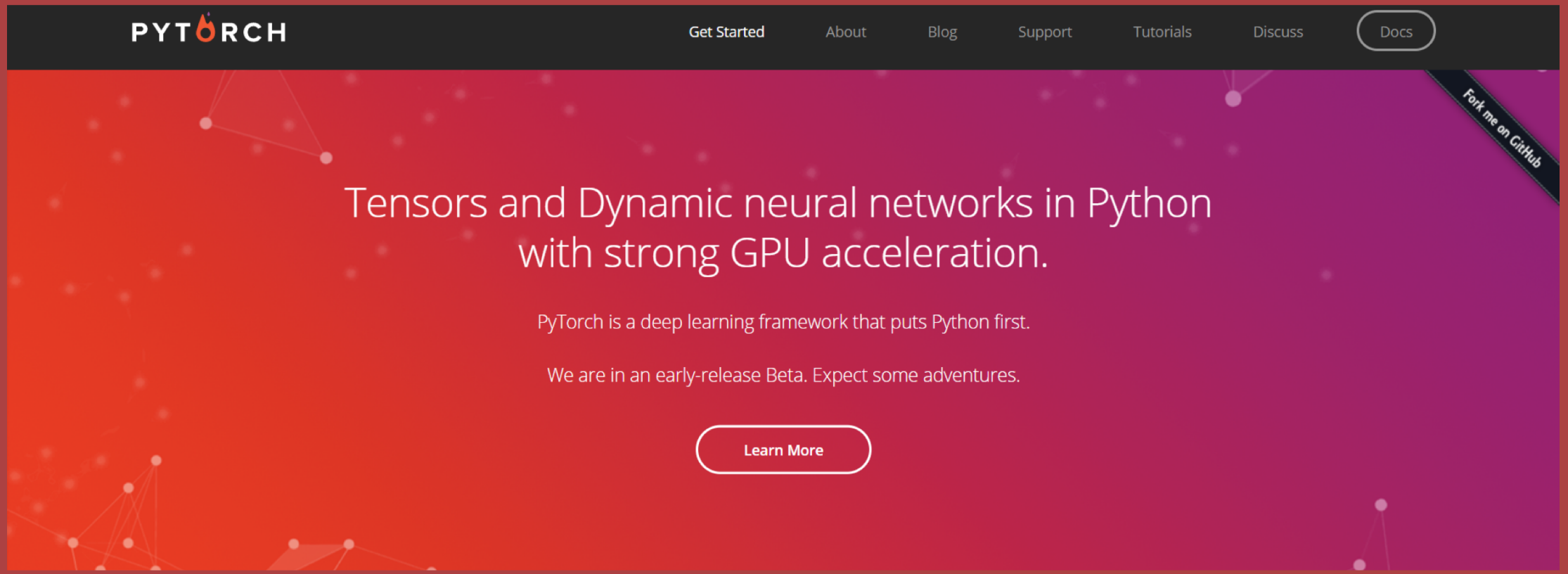
Eigen-based inference engine for OpenNMT models

Actual use cases:

- Transliteration in internal NLP library
- Demo translator on mobile phones

LIMITED GPU SUPPORT…

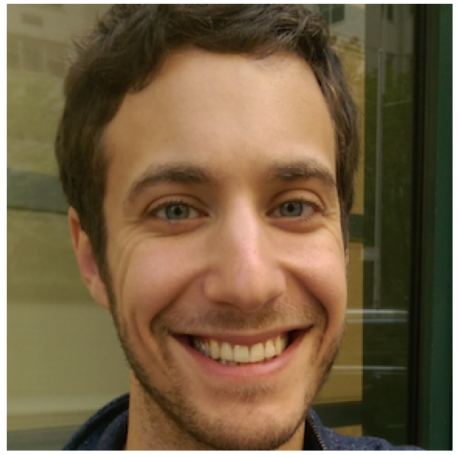
Chapter 4: Catching up with deep learning frameworks



PyTorch is released in February 2017

OpenNMT-py by PyTorch developers

Adam Lerer (Facebook)



Brian McCann (Metamind)

OpenNMT-py by OpenNMT

Growing list of features…

- Transformer
- Conv2Conv
- SRU cell
- Structured attention
- …

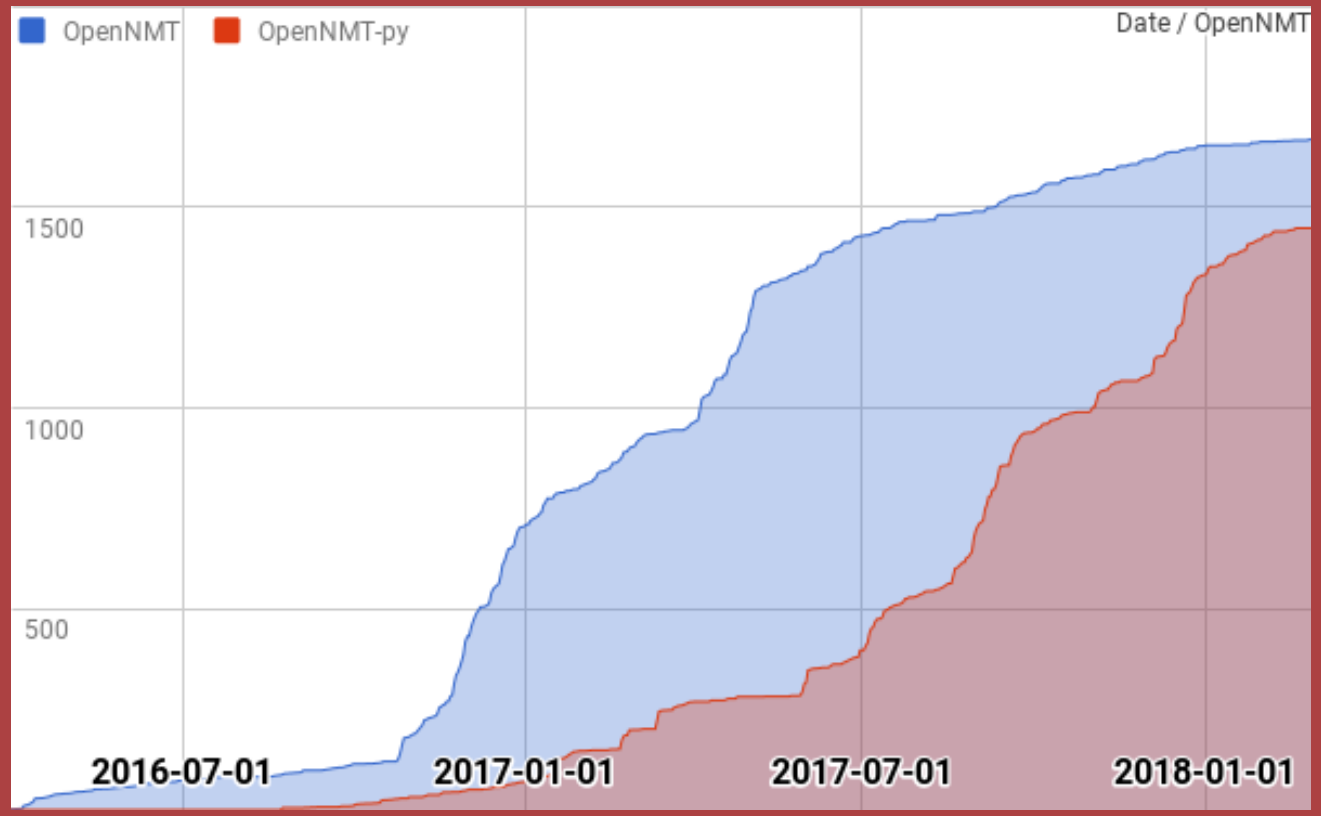
… thanks to many active users.


srush
235 commits 125,184 ++ 125,492 --


bpopeters
160 commits 1,798 ++ 1,522 --


JianyuZhan
146 commits 8,637 ++ 6,024 --


sebastianGehrmann
67 commits 1,252 ++ 650 --



Evolution of the number of commits in OpenNMT-py

Chapter 5: Diversifying the offer

OpenNMT-tf (November 2017)

README.md

build passing docs master chat on gitter

OpenNMT-tf

OpenNMT-tf is a general purpose sequence modeling tool in TensorFlow with production in mind. While neural machine translation is the main target task, it has been designed to more generally support:

- sequence to sequence mapping
- sequence tagging
- sequence classification

OpenNMT-tf: modular design

```python
"""Defines a medium-sized bidirectional LSTM encoder-decoder model."""

import tensorflow as tf
import opennmt as onmt

def model():
  return onmt.models.SequenceToSequence(
      source_inputter=onmt.inputters.WordEmbedder(
          vocabulary_file_key="source_words_vocabulary",
          embedding_size=512),
      target_inputter=onmt.inputters.WordEmbedder(
          vocabulary_file_key="target_words_vocabulary",
          embedding_size=512),
      encoder=onmt.encoders.BidirectionalRNNEncoder(
          num_layers=4,
          num_units=512,
          reducer=onmt.layers.ConcatReducer(),
          cell_class=tf.contrib.rnn.LSTMCell,
          dropout=0.3,
          residual_connections=False),
      decoder=onmt.decoders.AttentionalRNNDecoder(
          num_layers=4,
          num_units=512,
          bridge=onmt.layers.CopyBridge(),
          attention_mechanism_class=tf.contrib.seq2seq.LuongAttention,
          cell_class=tf.contrib.rnn.LSTMCell,
          dropout=0.3,
          residual_connections=False))
```

Generic building blocks:

- `ParallelInputter`
- `MixedInputter`

- `SequentialEncoder`
- `ParallelEncoder`

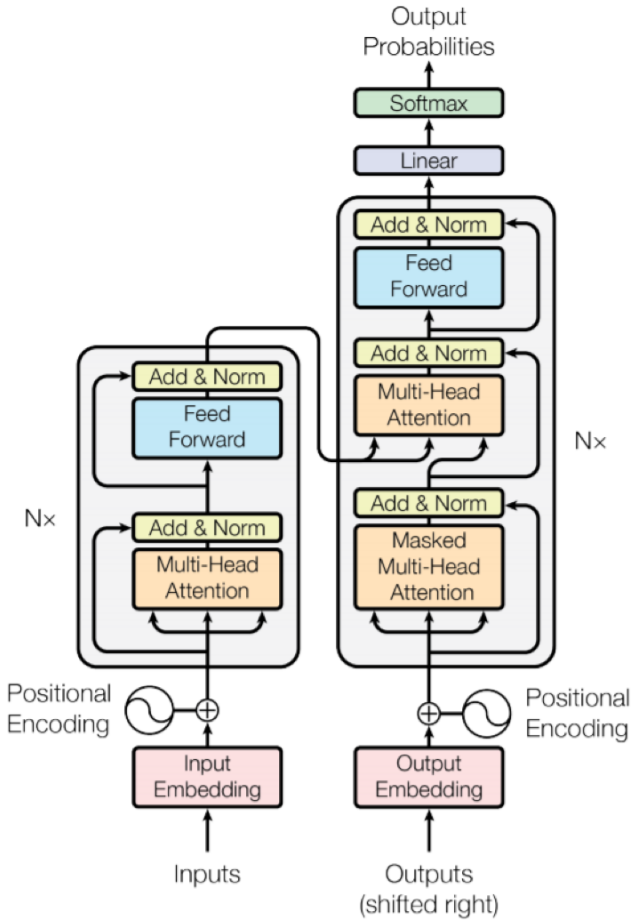
OpenNMT-tf: example of modular model

```
onmt.inputters.ParallelInputter(
    [
        onmt.inputters.MixedInputter(
            [
                onmt.inputters.WordEmbedder(...),
                onmt.inputters.CharConvEmbedder(...),
            ]),
        onmt.inputters.SequenceRecordInputter(...)
    ])
```

```
onmt.encoders.SequentialEncoder(
    [
        onmt.encoders.ParallelEncoder(
            [
                onmt.encoders.SelfAttentionEncoder(...),
                onmt.encoders.PyramidalRNNEncoder(...)
            ],
            reducer=onmt.layers.ConcatReducer(1)),

        onmt.encoders.ConvEncoder(...)
    ])
```

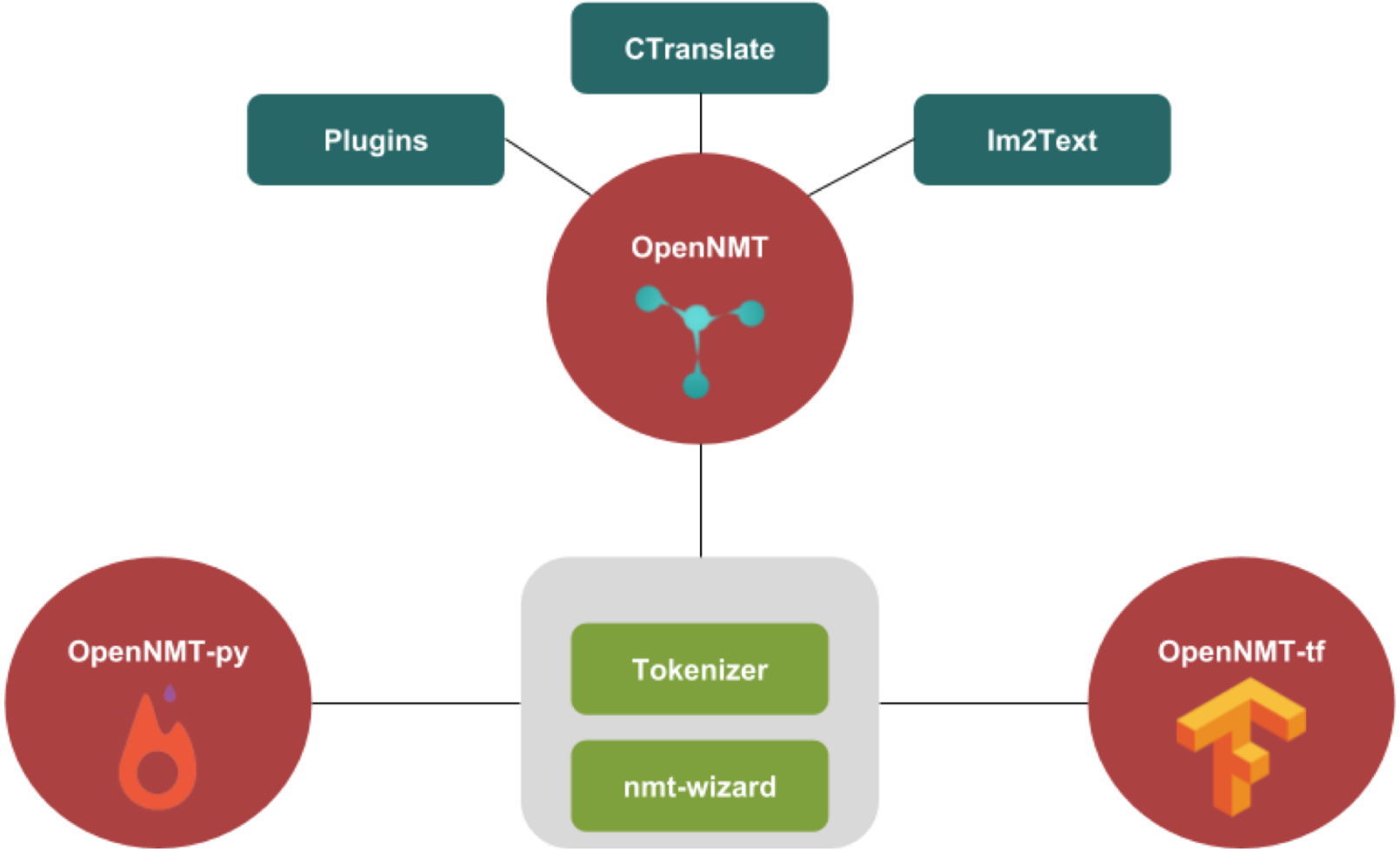
OpenNMT-tf: the Transformer adventure



- Several implementation and training details addressed over several weeks
- Strong alternative to *Tensor2Tensor* with model variants still to explore

Epilogue

Current OpenNMT landscape



Proposed direction

maintenance mode:

same level of support
but no major changes to
be expected

research-oriented:

flexible
hackable

production-oriented:

robust
clean APIs

Thank you!