

# 1 year developing and maintaining OpenNMT

Guillaume Klein

OpenNMT Workshop Paris, March 2018



## **... 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



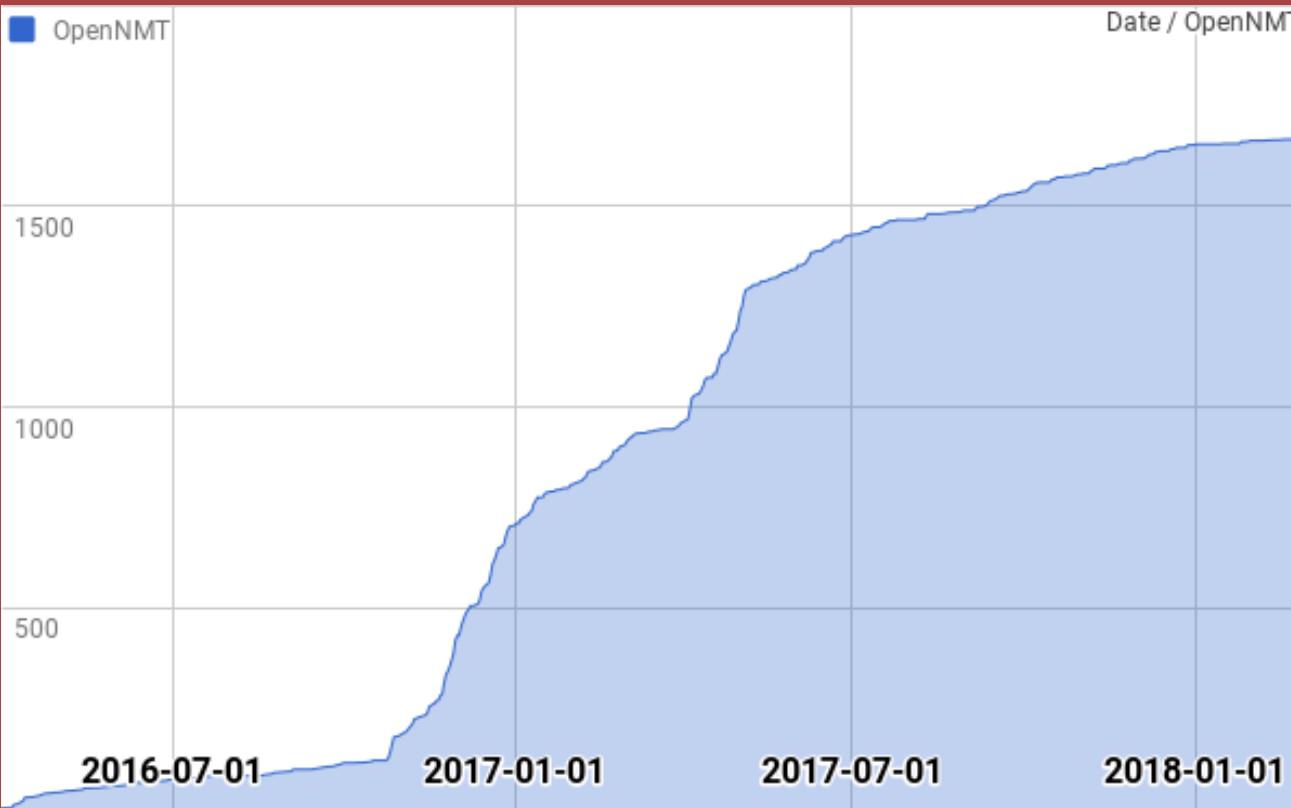
A screenshot of a GitHub repository page. The repository name is "harvardnlp / seq2seq-attn". The "Code" tab is selected. Other tabs include "Issues 7", "Pull requests 4", "Projects 1", and "Insights". Top navigation buttons include "Watch 80", "Star 902", "Fork 238", and a search bar. Below the tabs, a description reads "Sequence-to-sequence model with LSTM encoder/decoders and attention" followed by a link "http://nlp.seas.harvard.edu/code". A summary bar at the bottom shows "131 commits", "3 branches", "0 releases", "8 contributors", and the "MIT" license.

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	v0.1.0 (12/16)
New encoder types	v0.2.0 (01/17)
Multi GPU training	v0.3.0 (01/17)
Input vectors	v0.4.0 (02/17)
Translation servers	v0.5.0 (03/17)
Validation metrics	v0.6.0 (04/17)
Dynamic dataset	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 guarantees 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*

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 these features dictionaries:

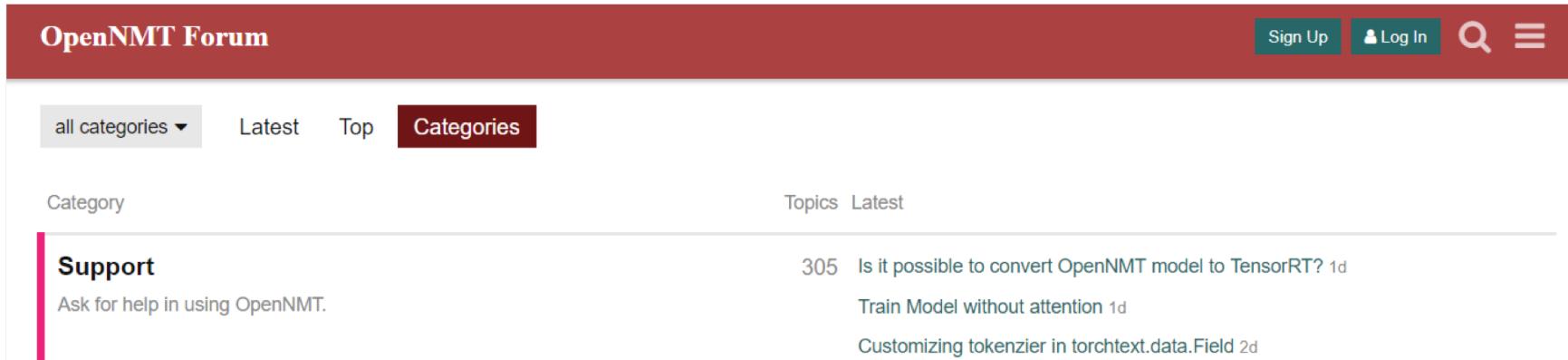
- `data/demo.source_feature_1.dict`
- `data/demo.source_feature_2.dict`
- `data/demo.source_feature_3.dict`

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

Table of contents

- Time-shifting
- Vocabularies
- Embeddings
- Beam search

**Goal:** answering and fixing issues fast.



The screenshot shows the OpenNMT Forum homepage. At the top, there is a red header bar with the text "OpenNMT Forum". On the right side of the header are buttons for "Sign Up", "Log In", a search icon, and a menu icon. Below the header, there is a navigation bar with links for "all categories ▾", "Latest", "Top", and a highlighted "Categories" button. To the right of the navigation bar, there are links for "Topics" and "Latest". The main content area has a table-like structure. The first column is labeled "Category" and contains a section titled "Support" with the sub-instruction "Ask for help in using OpenNMT.". The second column is labeled "Topics" and "Latest" and lists three topics: "305 Is it possible to convert OpenNMT model to TensorRT? 1d", "Train Model without attention 1d", and "Customizing tokenzier in torchtext.data.Field 2d".

# **Chapter 3: Extending the core project**



$$Q = (b + 1/b)\rho, \quad \rho = \frac{1}{2} \sum_{\alpha>0} \alpha,$$

The equation block contains two mathematical formulas. The first formula is  $Q = (b + 1/b)\rho$ , where each character is connected by a thin line to its corresponding LaTeX representation above it. The second formula is  $\rho = \frac{1}{2} \sum_{\alpha>0} \alpha$ , also with each character connected by lines to its LaTeX representation.

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

—Yuntian Deng

# OpenNMT/Tokenizer



OpenNMT / Tokenizer

Code Issues 0 Pull requests 0 Projects 0 Insights

Watch 8 Star 9 Fork 9

OpenNMT C++ tokenizer

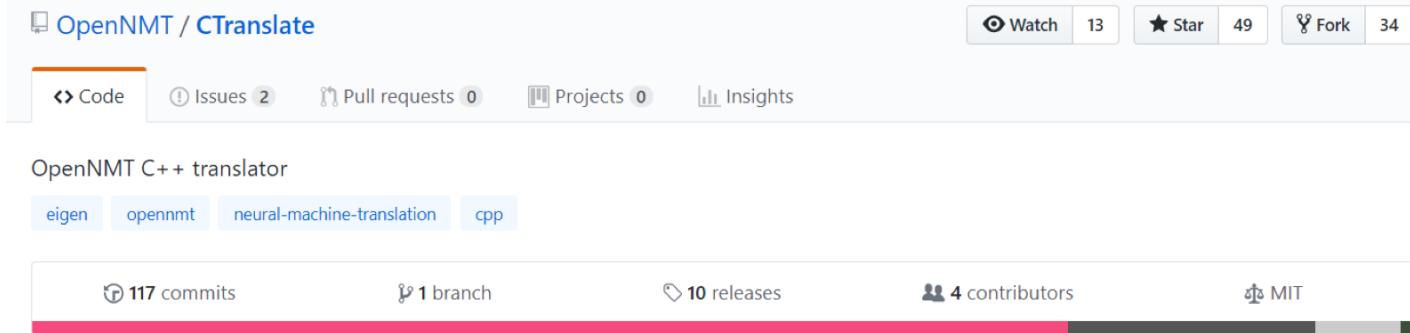
54 commits 2 branches 5 releases 5 contributors MIT

```
import pyonmttok

tokenizer = pyonmt.Tokenizer(
    mode=str,
    bpe_model_path="",
    joiner="\n",
    joiner_annotation=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.
```



*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**



[Get Started](#)[About](#)[Blog](#)[Support](#)[Tutorials](#)[Discuss](#)[Docs](#)

Fork me on GitHub

# Tensors and Dynamic neural networks in Python with strong GPU acceleration.

PyTorch is a deep learning framework that puts Python first.

We are in an early-release Beta. Expect some adventures.

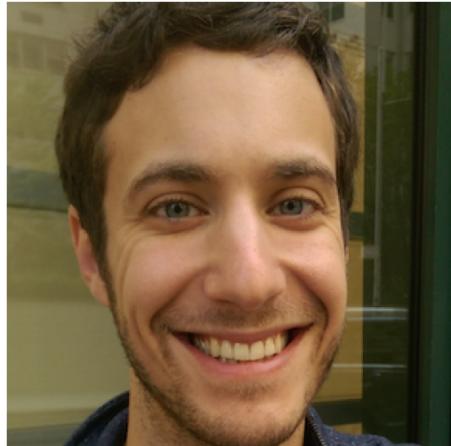
[Learn More](#)

## PyTorch is released in February 2017

# OpenNMT-py by PyTorch developers



**Adam Lerer (Facebook)**



**Brian McCann (Metamind)**

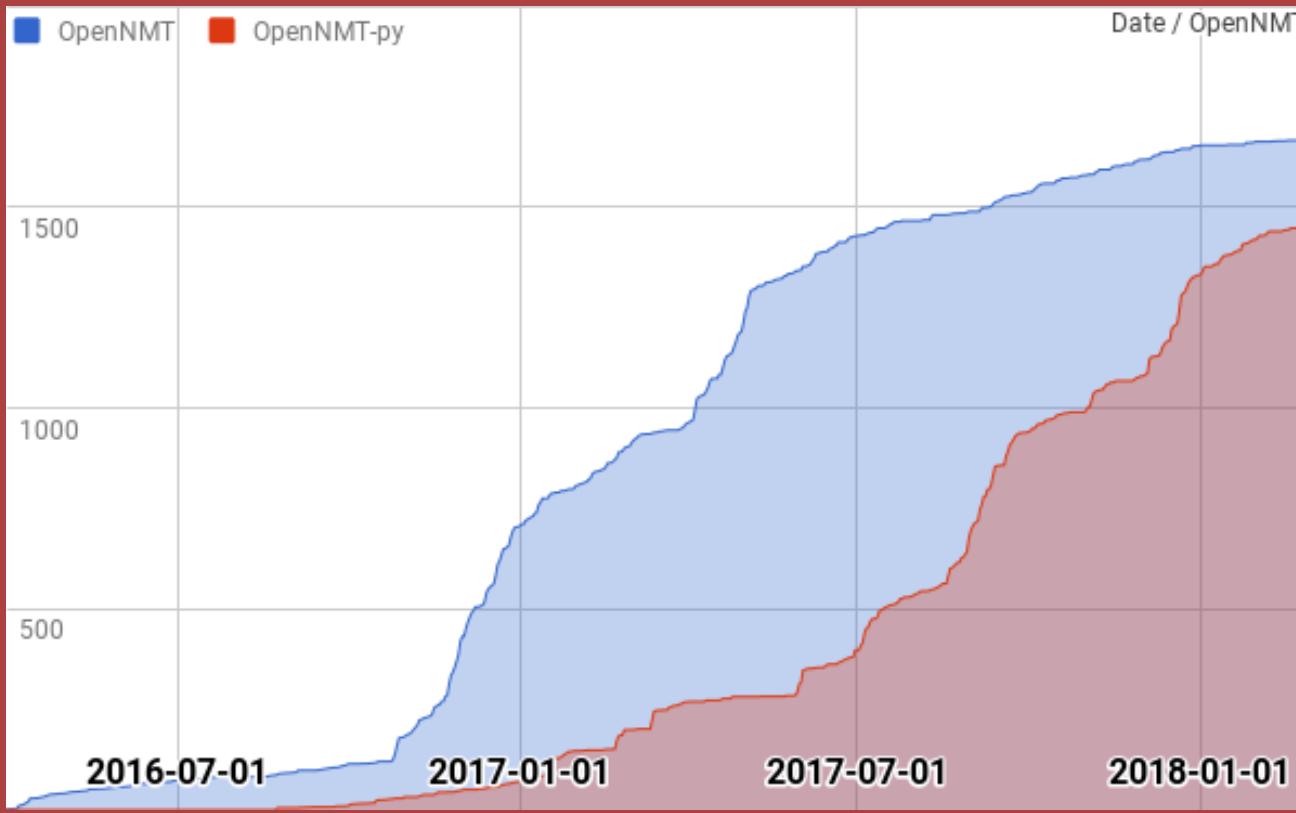


Growing list of features...

- Transformer
- Conv2Conv
- SRU cell
- Structured attention
- ...

... thanks to many active users.

	<a href="#">srush</a>
	235 commits 125,184 ++ 125,492 --
	<a href="#">bpeters</a>
	160 commits 1,798 ++ 1,522 --
	<a href="#">JianyuZhan</a>
	146 commits 8,637 ++ 6,024 --
	<a href="#">sebastianGehrmann</a>
	67 commits 1,252 ++ 650 --



Evolution of the number of commits in OpenNMT-py

# **Chapter 5: Diversifying the offer**



☰ 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



```
1 """Defines a medium-sized bidirectional LSTM encoder-decoder model."""
2
3 import tensorflow as tf
4 import opennmt as onmt
5
6 def model():
7     return onmt.models.SequenceToSequence(
8         source_inpointer=onmt.inputters.WordEmbedder(
9             vocabulary_file_key="source_words_vocabulary",
10            embedding_size=512),
11        target_inpointer=onmt.inputters.WordEmbedder(
12            vocabulary_file_key="target_words_vocabulary",
13            embedding_size=512),
14        encoder=onmt.encoders.BidirectionalRNNEncoder(
15            num_layers=4,
16            num_units=512,
17            reducer=onmt.layers.ConcatReducer(),
18            cell_class=tf.contrib.rnn.LSTMCell,
19            dropout=0.3,
20            residual_connections=False),
21        decoder=onmt.decoders.AttentionalRNNDecoder(
22            num_layers=4,
23            num_units=512,
24            bridge=onmt.layers.CopyBridge(),
25            attention_mechanism_class=tf.contrib.seq2seq.LuongAttention,
26            cell_class=tf.contrib.rnn.LSTMCell,
27            dropout=0.3,
28            residual_connections=False))
```

## Generic building blocks:

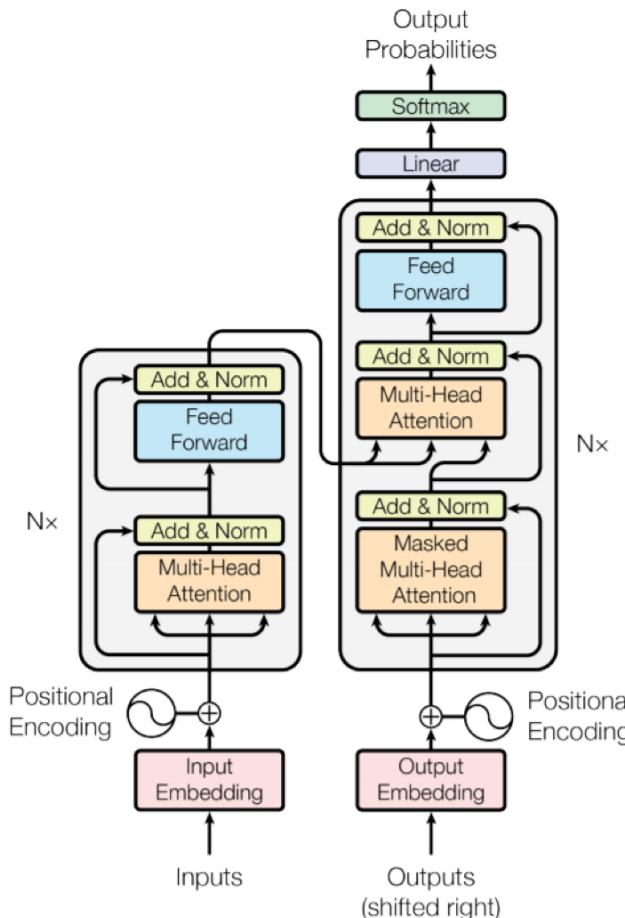
- **ParallelInpointer**
- **MixedInpointer**
- **SequentialEncoder**
- **ParallelEncoder**

# OpenNMT-tf: example of modular model



```
1 onmt.inputs.ParallelInputter(  
2     [  
3         onmt.inputs.MixedInputter(  
4             [  
5                 onmt.inputs.WordEmbedder(...),  
6                 onmt.inputs.CharConvEmbedder(...),  
7             ],  
8             onmt.inputs.SequenceRecordInputter(...)  
9         ])  
10    onmt.encoders.SequentialEncoder(  
11        [  
12            onmt.encoders.ParallelEncoder(  
13                [  
14                    onmt.encoders.SelfAttentionEncoder(...),  
15                    onmt.encoders.PyramidalRNNEncoder(...)  
16                ],  
17                reducer=onmt.layers.ConcatReducer(1)),  
18                onmt.encoders.ConvEncoder(...)  
19            ]))
```

# OpenNMT-tf: the Transformer adventure

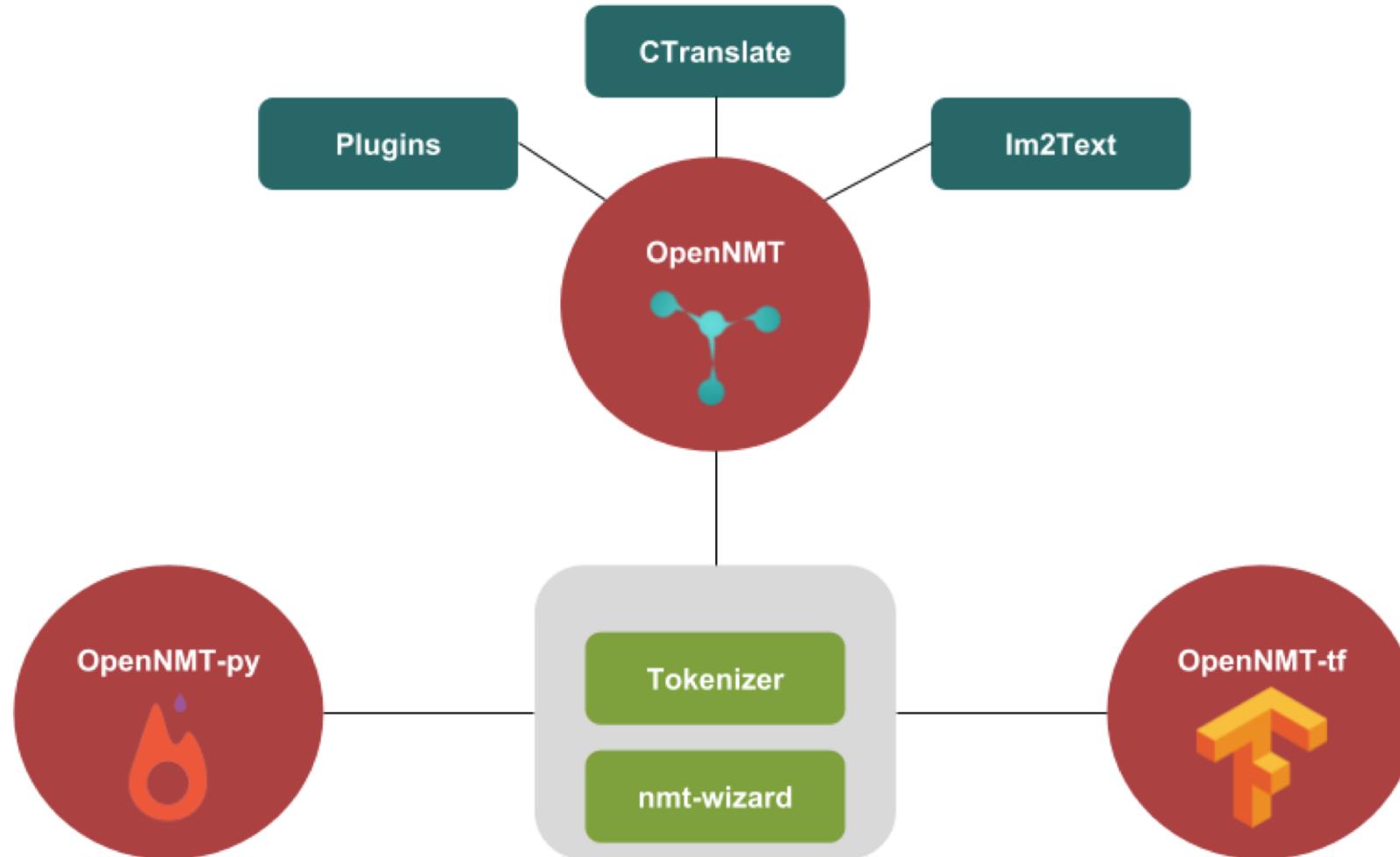


- Several implementation and training details addressed over several weeks
- Strong alternative to *Tensor2Tensor* with model variants still to explore

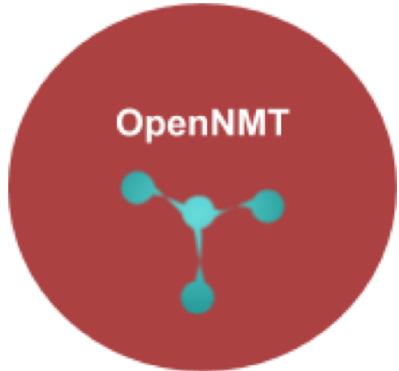
# Epilogue



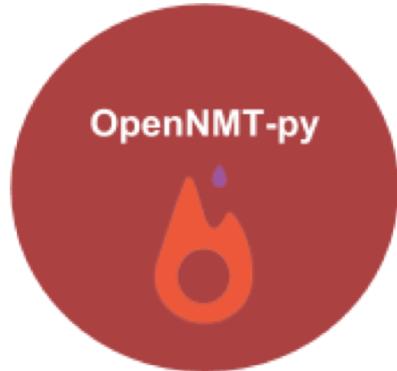
# Current OpenNMT landscape



# 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!

