# Contributors

OpenNMT-py is a community developed project and we love developer contributions.

## Guidelines

Before sending a PR, please do this checklist first:

- Please run `onmt/tests/pull\_request\_chk.sh` and fix any errors. When adding new functionality, also add tests to this script. Included checks:

1. flake8 check for coding style;

2. unittest;

3. continuous integration tests listed in `.travis.yml`.

- When adding/modifying class constructor, please make the arguments as same naming style as its superclass in PyTorch.

- If your change is based on a paper, please include a clear comment and reference in the code (more on that below).

### Docstrings

Above all, try to follow the Google docstring format

([Napoleon example](https://sphinxcontrib-napoleon.readthedocs.io/en/latest/example\_google.html),

[Google styleguide](http://google.github.io/styleguide/pyguide.html)).

This makes it easy to include your contributions in the Sphinx documentation. And, do feel free

to autodoc your contributions in the API ``.rst`` files in the `docs/source` folder! If you do, check that

your additions look right.

```bash

cd docs

# install some dependencies if necessary:

# recommonmark, sphinx\_rtd\_theme, sphinxcontrib-bibtex

make html

firefox build/html/main.html # or your browser of choice

```

Some particular advice:

- Try to follow Python 3 [``typing`` module](https://docs.python.org/3/library/typing.html) conventions when documenting types.

- Exception: use "or" instead of unions for more readability

- For external types, use the full "import name". Common abbreviations (e.g. ``np``) are acceptable.

For ``torch.Tensor`` types, the ``torch.`` is optional.

- Please don't use tics like `` (`str`) `` or rst directives like `` (:obj:`str`) ``. Napoleon handles types

very well without additional help, so avoid the clutter.

- [Google docstrings don't support multiple returns](https://stackoverflow.com/questions/29221551/can-sphinx-napoleon-document-function-returning-multiple-arguments).

For multiple returns, the following works well with Sphinx and is still very readable.

```python

def foo(a, b):

"""This is my docstring.

Args:

a (object): Something.

b (class): Another thing.

Returns:

(object, class):

\* a: Something or rather with a long

description that spills over.

\* b: And another thing.

"""

return a, b

```

- When citing a paper, avoid directly linking in the docstring! Add a Bibtex entry to `docs/source/refs.bib`.

E.g., to cite "Attention Is All You Need", visit [arXiv](https://arxiv.org/abs/1706.03762), choose the

[bibtext](https://dblp.uni-trier.de/rec/bibtex/journals/corr/VaswaniSPUJGKP17) link, search `docs/source/refs.bib`

using `CTRL-F` for `DBLP:journals/corr/VaswaniSPUJGKP17`, and if you do not find it then copy-paste the

citation into `refs.bib`. Then, in your docstring, use ``:cite:`DBLP:journals/corr/VaswaniSPUJGKP17` ``.

- However, a link is better than nothing.

- Please document tensor shapes. Prefer the format

``` ``(a, b, c)`` ```. This style is easy to read, allows using ``x`` for multplication, and is common

(PyTorch uses a few variations on the parentheses format, AllenNLP uses exactly this format, Fairseq uses

the parentheses format with single ticks).

- Again, a different style is better than no shape documentation.

- Please avoid unnecessary space characters, try to capitalize, and try to punctuate.

For multi-line docstrings, add a blank line after the closing ``"""``.

Don't use a blank line before the closing quotes.

``""" not this """`` ``"""This."""``

```python

"""

Not this.

"""

```

```python

"""This."""

```

This note is the least important. Focus on content first, but remember that consistent docs look good.

- Be sensible about the first line. Generally, one stand-alone summary line (per the Google guidelines) is good.

Sometimes, it's better to cut directly to the args or an extended description. It's always acceptable to have a

"trailing" citation.