# How to contribute to transformers?

Everyone is welcome to contribute, and we value everybody's contribution. Code

is thus not the only way to help the community. Answering questions, helping

others, reaching out and improving the documentations are immensely valuable to

the community.

It also helps us if you spread the word: reference the library from blog posts

on the awesome projects it made possible, shout out on Twitter every time it has

helped you, or simply star the repo to say "thank you".

## You can contribute in so many ways!

There are 4 ways you can contribute to transformers:

\* Fixing outstanding issues with the existing code;

\* Implementing new models;

\* Contributing to the examples or to the documentation;

\* Submitting issues related to bugs or desired new features.

\*All are equally valuable to the community.\*

## Submitting a new issue or feature request

Do your best to follow these guidelines when submitting an issue or a feature

request. It will make it easier for us to come back to you quickly and with good

feedback.

### Did you find a bug?

The transformers are robust and reliable thanks to the users who notify us of

the problems they encounter. So thank you for reporting an issue.

First, we would really appreciate it if you could \*\*make sure the bug was not

already reported\*\* (use the search bar on Github under Issues).

Did not find it? :( So we can act quickly on it, please follow these steps:

\* Include your \*\*OS type and version\*\*, the versions of \*\*Python\*\*, \*\*PyTorch\*\* and

\*\*Tensorflow\*\* when applicable;

\* A short, self-contained, code snippet that allows us to reproduce the bug in

less than 30s;

\* Provide the \*full\* traceback if an exception is raised.

To get the OS and software versions automatically, you can run the following command:

```bash

transformers-cli env

```

or from the root of the repository the following command:

```bash

python src/transformers/commands/transformers\_cli.py env

```

### Do you want to implement a new model?

Awesome! Please provide the following information:

\* Short description of the model and link to the paper;

\* Link to the implementation if it is open-source;

\* Link to the model weights if they are available.

If you are willing to contribute the model yourself, let us know so we can best

guide you.

We have added a \*\*detailed guide and templates\*\* to guide you in the process of adding a new model. You can find them

in the [`templates`](https://github.com/huggingface/transformers/tree/master/templates) folder.

### Do you want a new feature (that is not a model)?

A world-class feature request addresses the following points:

1. Motivation first:

\* Is it related to a problem/frustration with the library? If so, please explain

why. Providing a code snippet that demonstrates the problem is best.

\* Is it related to something you would need for a project? We'd love to hear

about it!

\* Is it something you worked on and think could benefit the community?

Awesome! Tell us what problem it solved for you.

2. Write a \*full paragraph\* describing the feature;

3. Provide a \*\*code snippet\*\* that demonstrates its future use;

4. In case this is related to a paper, please attach a link;

5. Attach any additional information (drawings, screenshots, etc.) you think may help.

If your issue is well written we're already 80% of the way there by the time you

post it.

We have added \*\*templates\*\* to guide you in the process of adding a new example script for training or testing the

models in the library. You can find them in the [`templates`](https://github.com/huggingface/transformers/tree/master/templates)

folder.

## Start contributing! (Pull Requests)

Before writing code, we strongly advise you to search through the exising PRs or

issues to make sure that nobody is already working on the same thing. If you are

unsure, it is always a good idea to open an issue to get some feedback.

You will need basic `git` proficiency to be able to contribute to

`transformers`. `git` is not the easiest tool to use but it has the greatest

manual. Type `git --help` in a shell and enjoy. If you prefer books, [Pro

Git](https://git-scm.com/book/en/v2) is a very good reference.

Follow these steps to start contributing:

1. Fork the [repository](https://github.com/huggingface/transformers) by

clicking on the 'Fork' button on the repository's page. This creates a copy of the code

under your GitHub user account.

2. Clone your fork to your local disk, and add the base repository as a remote:

```bash

$ git clone git@github.com:<your Github handle>/transformers.git

$ cd transformers

$ git remote add upstream https://github.com/huggingface/transformers.git

```

3. Create a new branch to hold your development changes:

```bash

$ git checkout -b a-descriptive-name-for-my-changes

```

\*\*do not\*\* work on the `master` branch.

4. Set up a development environment by running the following command in a virtual environment:

```bash

$ pip install -e ".[dev]"

```

(If transformers was already installed in the virtual environment, remove

it with `pip uninstall transformers` before reinstalling it in editable

mode with the `-e` flag.)

Right now, we need an unreleased version of `isort` to avoid a

[bug](https://github.com/timothycrosley/isort/pull/1000):

```bash

$ pip install -U git+git://github.com/timothycrosley/isort.git@e63ae06ec7d70b06df9e528357650281a3d3ec22#egg=isort

```

5. Develop the features on your branch.

As you work on the features, you should make sure that the test suite

passes:

```bash

$ make test

```

`transformers` relies on `black` and `isort` to format its source code

consistently. After you make changes, format them with:

```bash

$ make style

```

`transformers` also uses `flake8` to check for coding mistakes. Quality

control runs in CI, however you can also run the same checks with:

```bash

$ make quality

```

Once you're happy with your changes, add changed files using `git add` and

make a commit with `git commit` to record your changes locally:

```bash

$ git add modified\_file.py

$ git commit

```

Please write [good commit

messages](https://chris.beams.io/posts/git-commit/).

It is a good idea to sync your copy of the code with the original

repository regularly. This way you can quickly account for changes:

```bash

$ git fetch upstream

$ git rebase upstream/master

```

Push the changes to your account using:

```bash

$ git push -u origin a-descriptive-name-for-my-changes

```

6. Once you are satisfied (\*\*and the checklist below is happy too\*\*), go to the

webpage of your fork on GitHub. Click on 'Pull request' to send your changes

to the project maintainers for review.

7. It's ok if maintainers ask you for changes. It happens to core contributors

too! So everyone can see the changes in the Pull request, work in your local

branch and push the changes to your fork. They will automatically appear in

the pull request.

### Checklist

1. The title of your pull request should be a summary of its contribution;

2. If your pull request adresses an issue, please mention the issue number in

the pull request description to make sure they are linked (and people

consulting the issue know you are working on it);

3. To indicate a work in progress please prefix the title with `[WIP]`. These

are useful to avoid duplicated work, and to differentiate it from PRs ready

to be merged;

4. Make sure existing tests pass;

5. Add high-coverage tests. No quality testing = no merge.

- If you are adding a new model, make sure that you use

`ModelTester.all\_model\_classes = (MyModel, MyModelWithLMHead,...)`, which triggers the common tests.

- If you are adding new `@slow` tests, make sure they pass using

`RUN\_SLOW=1 python -m pytest tests/test\_my\_new\_model.py`.

- If you are adding a new tokenizer, write tests, and make sure

`RUN\_SLOW=1 python -m pytest tests/test\_tokenization\_{your\_model\_name}.py` passes.

CircleCI does not run the slow tests, but github actions does every night!

6. All public methods must have informative docstrings that work nicely with sphinx. See `modeling\_ctrl.py` for an

example.

### Tests

An extensive test suite is included to test the library behavior and several examples. Library tests can be found in

the [tests folder](https://github.com/huggingface/transformers/tree/master/tests) and examples tests in the

[examples folder](https://github.com/huggingface/transformers/tree/master/examples).

We like `pytest` and `pytest-xdist` because it's faster. From the root of the

repository, here's how to run tests with `pytest` for the library:

```bash

$ python -m pytest -n auto --dist=loadfile -s -v ./tests/

```

and for the examples:

```bash

$ pip install -r examples/requirements.txt # only needed the first time

$ python -m pytest -n auto --dist=loadfile -s -v ./examples/

```

and for the slow tests:

```bash

RUN\_SLOW=yes python -m pytest -n auto --dist=loadfile -s -v ./tests/

```

or

```python

RUN\_SLOW=yes python -m pytest -n auto --dist=loadfile -s -v ./tests/

```

In fact, that's how `make test` and `make test-examples` are implemented!

You can specify a smaller set of tests in order to test only the feature

you're working on.

By default, slow tests are skipped. Set the `RUN\_SLOW` environment variable to

`yes` to run them. This will download many gigabytes of models make sure you

have enough disk space and a good Internet connection, or a lot of patience!

```bash

$ RUN\_SLOW=yes python -m pytest -n auto --dist=loadfile -s -v ./tests/

$ RUN\_SLOW=yes python -m pytest -n auto --dist=loadfile -s -v ./examples/

```

Likewise, set the `RUN\_CUSTOM\_TOKENIZERS` environment variable to `yes` to run

tests for custom tokenizers, which don't run by default either.

? Transformers uses `pytest` as a test runner only. It doesn't use any

`pytest`-specific features in the test suite itself.

This means `unittest` is fully supported. Here's how to run tests with

`unittest`:

```bash

$ python -m unittest discover -s tests -t . -v

$ python -m unittest discover -s examples -t examples -v

```

### Style guide

For documentation strings, `transformers` follows the [google style](https://google.github.io/styleguide/pyguide.html).

Check our [documentation writing guide](https://github.com/huggingface/transformers/tree/master/docs#writing-documentation---specification)

for more information.

#### This guide was heavily inspired by the awesome [scikit-learn guide to contributing](https://github.com/scikit-learn/scikit-learn/blob/master/CONTRIBUTING.md)