How To Request Support

This is an Open Source Project so please be mindful that like in any other project of this kind there is no obligation to answer all requests for help.

However, we want to encourage you to ask for help whenever you think it's needed! We are happy about every question we get because it allows us to better understand your needs, possible misunderstandings, and most importantly a way for you to help us make this library better. That being said, this document's main purpose is to provide guidelines at how you can formulate your requests to increase your chances to be understood and to get support.

There are two main venues to receive support: the forums and the GitHub issues.

The Forums

The user forums are supported by the wide community of the library users and backed up by developers when needed.

If you have a difficulty with deploying this library or some questions, or you'd like to discuss a new feature, please first consider discussing those things at the forums. Only when you feel your subject matter has been crystalized and you still need support from the library developers do proceed to file an issue.

In particular all "Please explain" questions or objectively very user-specific feature requests belong to the forums. Here are some example of such questions:

- "I would like to use a BertModel within a RL-Agent for a customer support service. How can I use a BertForMaskedLM in my ChatBotModel?"
- "Could you please explain why T5 has no positional embedding matrix under T5Model?"
- "How should I set my generation parameters for translation?"
- "How to train T5 on De->En translation?"

The GitHub Issues

Everything which hints at a bug should be opened as an issue.

You are not required to read the following guidelines before opening an issue. However, if you notice that your issue doesn't get any replies, chances are that the developers have one or several difficulties with its quality. In this case, reading the following points and adjusting your issue accordingly could help.

1. Before posting an issue, first search for already posted issues, since chances are someone has already asked a similar question before you.

   If you use Google your search query should be:

   ```
   "huggingface" "transformers" your query
   ```

   The first two quoted words tell Google to limit the search to the context of the Huggingface Transformers. The remainder is your query - most commonly this would be the error message the software fails with. We will go deeper into details shortly.

The results of such a query will typically match GitHub issues, Hugging Face forums, StackExchange, and blogs.

If you find relevant hints, you may choose to continue the discussion there if you have follow up questions.

If what you found is similar but doesn't quite answer your problem, please, post a new issue and do include links to similar issues or forum discussions you may have found.

Let's look at some examples:

The error message, often referred to as an assertion, tells us what went wrong. Here is an example of an assertion:

```
Traceback (most recent call last):
  File "<string>", line 1, in <module>
  File "/transformers/src/transformers/__init__.py", line 34, in <module>
    from . import dependency_versions_check
  File "/transformers/src/transformers/dependency_versions_check.py", line
34, in <module>
    from .utils import is_tokenizers_available
  File "/transformers/src/transformers/utils/import_utils.py", line 40, in
<module>
    from tqdm.auto import tqdm
 ModuleNotFoundError: No module named 'tqdm.auto'
```

and it typically includes a traceback, so that we can see the full stack of calls the program made before it fails. This gives us the context to know why the program failed.

Going back to the above example. If you received this error search, look at the very last line of the error which is:

```
ModuleNotFoundError: No module named 'tqdm.auto'
```

And now we can use it to do the searching on your favorite search engine:

1. first for `"huggingface" "transformers" "ModuleNotFoundError: No module named 'tqdm.auto'"`
2. if you don't find relevant results, then search for just `"ModuleNotFoundError: No module named 'tqdm.auto'"`
3. and finally if nothing still comes up, then remove the outside quotes: `ModuleNotFoundError: No module named 'tqdm.auto'`

If the error includes any messages that include bits unique to your filesystem, always remove those in the search query since other users will not have the same filesystem as yours. For example:

```
python -c 'open("/tmp/wrong_path.txt", "r")'
Traceback (most recent call last):
  File "<string>", line 1, in <module>
FileNotFoundError: [Errno 2] No such file or directory: '/tmp/wrong_path.txt'
```

Here you'd search for just: `"FileNotFoundError: [Errno 2] No such file or directory"`

If the local information that you removed were inside the error message and you removed them you may need to remove double quotes since your query is no longer exact. So if the error message was something like:

```
    ValueError: '/tmp/wrong_path.txt' cannot be found
```

then you'd search for `"ValueError" "cannot be found"`

As you search you will notice that when you don't use quotes often the search engines will return a variety of unrelated hits, which may or may not be what you want.

Experiment with different ways and find which approach gives the most satisfactory results.

2. Keep the issue short, providing the information that you think will aid the developers to understand your situation. Put yourself in the shoes of the person who has never seen your code or knows anything about your custom setup. This mental exercise will help to develop an intuition to what/what not to share"

3. If there is a software failure, always provide the full traceback, for example:

```
$ python -c 'import transformers'
Traceback (most recent call last):
  File "<string>", line 1, in <module>
  File "/transformers/src/transformers/__init__.py", line 34, in <module>
    from . import dependency_versions_check
  File "/transformers/src/transformers/dependency_versions_check.py", line
34, in <module>
    from .utils import is_tokenizers_available
  File "/transformers/src/transformers/utils/import_utils.py", line 40, in
<module>
    from tqdm.auto import tqdm
ModuleNotFoundError: No module named 'tqdm.auto'
```

As compared to providing just the last line of the error message, e.g.:

```
ModuleNotFoundError: No module named 'tqdm.auto'
```

which is not sufficient.

If your application is running on more than one GPU (e.g. under `DistributedDataParallel`) and typically getting every log and traceback printed multiple times, please make sure that you paste only one copy of it. At times the traceback from parallel processes may get interleaved - so either disentangle these or change the loggers to log only for `local_rank==0` so that only one process logs things.

4. When quoting a traceback, command line instructions and any type of code always enclose it in triple backticks inside the editor window, that is:

````
```
git clone https://github.com/huggingface/transformers
cd transformers
pip install .
```
````

If it's a command line with a long argument list, please consider breaking it down using backslashes and new lines. Here is an example of a good command line quote:

```
 cd examples/seq2seq
 python -m torch.distributed.launch --nproc_per_node=2 ./finetune_trainer.py
\
 --model_name_or_path sshleifer/distill-mbart-en-ro-12-4 --data_dir wmt_en_ro
\
 --output_dir output_dir --overwrite_output_dir \
 --do_train --n_train 500 --num_train_epochs 1 \
 --per_device_train_batch_size 1  --freeze_embeds \
 --src_lang en_XX --tgt_lang ro_RO --task translation \
 --fp16 --sharded_ddp
```

If you don't break it up, one has to scroll horizontally which often makes it quite difficult to quickly see what's happening.

The backslashes allow us to copy the command directly into the console to run it, without needing to edit it.

5. Include only the important information that you think will help the developer to quickly identify the problem.

   For example applications often create huge amounts of logs. Ask yourself whether providing all or parts of the log is useful.

   Pasting a 100-1000 lines of log into the issue is an immediate turn off, since it will take a lot of time to figure out where the pertinent parts of the log are.

   Attaching a full log can be helpful if it's done as an attachment, if it's enclosed in the following html code in the comment editor window:

   ```
   <details>
   <summary>Full log</summary>
   <pre>

   many
   lines
   go
   here

   </pre>
   </details>
   ```

   which would result in the following entry, which can be opened if desired, but otherwise takes little space.

   ▶ Full log

   You could also provide a link to a pastebin service, but this is less beneficial since those links tend to expire quickly and future readers of your issue might not be able to access that log file anymore and may lack some context.

6. If this is an issue in your code, do try to reduce that code to a minimal example that still demonstrates the problem. Please ask at the forums if you have a hard time figuring how to do that. Please realize that we don't have the luxury of having time to try and understand all of your custom code.

If you really tried to make a short reproducible code but couldn't figure it out, it might be that having a traceback will give the developer enough information to know what's going on. But if it is not enough and we can't reproduce the problem, we can't really solve it.

Do not despair if you can't figure it out from the beginning, just share what you can and perhaps someone else will be able to help you at the forums.

If your setup involves any custom datasets, the best way to help us reproduce the problem is to create a Google Colab notebook that demonstrates the issue and once you verify that the issue still exists, include a link to that notebook in the Issue. Just make sure that you don't copy and paste the location bar url of the open notebook - as this is private and we won't be able to open it. Instead, you need to click on `Share` in the right upper corner of the notebook, select `Get Link` and then copy and paste the public link it will give to you.

7. If you forked off some of this project's code or example applications, please, do not ask us to go into your code repository and figure out what you may have done. The code is already very complex and unless there is an easy way to do a diff and it's a small diff, it won't be possible to find someone with time on their hands to make a lengthy investigation. Albeit, you might find someone at the forums who will be generous to do this for you.

8. Before reporting an issue, first, always try to update your environment to the latest official version of this library. We have no resources to go and debug older revisions, which could easily have bugs that have been fixed in the latest released version.

   We understand that this is not always possible, especially when APIs change, in which case file an issue against the highest library version your environment can support.

   Of course, if you upgrade the library, always retest that the problem is still there.

9. Please do not ask us to reproduce an issue with your custom data, since we don't have it. So, either you should use some existing dataset supported by HF datasets or you need to supply a code that generates a small sample on the fly, or some another quick and simple way to get it.

   Please do not send us any non-public domain data that may require a license or a permission to be used.

10. Do not tag multiple developers on the issue unless you know this is expected, either because you asked them and they gave you an explicit permission to tag them or the issue template instructs you to do so.

The "who to tag for what domain" part of the issue template is there to help users direct their questions to the right developers who are designated maintainers of project's specific domains. They can then decide at their own discretion to tag other developers if they feel it'd help move the issue forward.

We currently don't have a triage service and we trust your capacity to identify the right domain and thus the persons to tag in your issue. If you are not sure, please use the forums to ask for guidance.

When in doubt, err on the side of not tagging a given person. If you tag multiple people out of context or permission don't be surprised if you get no response at all. Please remember that every time you tag someone, they get a notification and you're taking their time without their permission. Please be sensitive to that.

If you got helped by one of the developers in the past please don't tag them in future issues, unless they are listed in the issue template for the domain you are asking about or that developer gave you an explicit permission to tag them in future issues.

If you see a certain developer doing multiple and/or recent commits into a specific area of the project that you feel is relevant to your issue, it is not a good reason to tag them. Various developers may be fixing things that prevent them

from moving forward, but often their work is focused on a totally different domain. And while they may or may not know how to help you with the problem at hand, it would benefit the whole community much more if they focus on the domain of their unique expertise.

11. Use the Edit button. Take your time, and re-read and improve the wording and formatting to make your posts and comments as easy to understand as possible.

    Avoid posting multiple comments in a row, as each comment generates a notification for the developers tagged in that issue. If you happened to post multiple comments in a row, and nobody followed up yet - consider merging those into one or a few comments while editing the combined content to be coherent.

    If you choose to edit your older comments after others posted follow up comments you need to be aware that your modifications might not be noticed, so if it's not a typo fixing, try to write a new comment flagging that something has been changed in the previous comments.

    For example, the very first comment is the most important one. If while the thread unfolds you realize that things aren't as they seemed to you originally you may want to edit the first post to reflect the up-to-date understanding of the issue at hand so that it helps those who read your issue in the future quickly understand what's going on and not need to sift through dozens of comments. It also helps to indicate that the post was edited. So, those reading the thread later can understand why there might be certain discontinuity in the information flow.

    Use bullets and items if you have lists of items and the outcome improves overall readability.

    Use backticks to refer to class and function names, e.g. `BartModel` and `generate` as these stand out and improve the speed of a reader's comprehension.

    Try not use italics and bold text too much as these often make the text more difficult to read.

12. If you are cross-referencing a specific comment in a given thread or another issue, always link to that specific comment, rather than using the issue link. If you do the latter it could be quite impossible to find which specific comment you're referring to.

    To get the link to the specific comment do not copy the url from the location bar of your browser, but instead, click the `...` icon in the upper right corner of the comment and then select "Copy Link".

    For example the first link is a link to an issue, and the second to a specific comment in the same issue:

    1. https://github.com/huggingface/transformers/issues/9257
    2. https://github.com/huggingface/transformers/issues/9257#issuecomment-749945162

13. If you are replying to a last comment, it's totally fine to make your reply with just your comment in it. The readers can follow the information flow here.

    But if you're replying to a comment that happened some comments back it's always a good practice to quote just the relevant lines you're replying it. The `>` is used for quoting, or you can always use the menu to do so. For example your editor box will look like:

    ```
    > How big is your gpu cluster?

    Our cluster is made of 256 gpus.
    ```

    If you are addressing multiple comments, quote the relevant parts of each before your answer. Some people use the same comment to do multiple replies, others separate them into separate comments. Either way works. The latter approach helps for linking to a specific comment.

In general the best way to figure out what works the best is learn from issues posted by other people - see which issues get great responses and which get little to no response - observe what the posters who received great responses did differently from those who did not.

Thank you for reading this somewhat lengthy document. We would like to conclude that these are not absolute rules, but a friendly advice that will help maximize the chances for us to understand what you are trying to communicate, reproduce the problem then resolve it to your satisfaction and the benefit of the whole community.

If after reading this document there are remaining questions on how and why or there is a need for further elucidation, please, don't hesitate to ask your question in this thread.