If you're opening this Notebook on colab, you will probably need to install 🤗 Transformers and 🤗 Datasets as well as other dependencies. Uncomment the following cell and run it.

```
! pip install datasets evaluate transformers rouge-score nltk sentencepiece
```

```
Requirement already satisfied: datasets in /usr/local/lib/python3.10/dist-packages (2.14.5)
Requirement already satisfied: evaluate in /usr/local/lib/python3.10/dist-packages (0.4.0)
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-packages (4.34.0)
Requirement already satisfied: rouge-score in /usr/local/lib/python3.10/dist-packages (0.1.2)
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: sentencepiece in /usr/local/lib/python3.10/dist-packages (0.1.99)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from datasets) (1.23.5)
Requirement already satisfied: pyarrow>=8.0.0 in /usr/local/lib/python3.10/dist-packages (from datasets) (9.0.0)
Requirement already satisfied: dill<0.3.8,>=0.3.0 in /usr/local/lib/python3.10/dist-packages (from datasets) (0.3.7)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from datasets) (1.5.3)
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.10/dist-packages (from datasets) (2.31.0)
Requirement already satisfied: tqdm>=4.62.1 in /usr/local/lib/python3.10/dist-packages (from datasets) (4.66.1)
Requirement already satisfied: xxhash in /usr/local/lib/python3.10/dist-packages (from datasets) (3.4.1)
Requirement already satisfied: multiprocess in /usr/local/lib/python3.10/dist-packages (from datasets) (0.70.15)
Requirement already satisfied: fsspec[http]<2023.9.0,>=2023.1.0 in /usr/local/lib/python3.10/dist-packages (from dataset
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packages (from datasets) (3.8.5)
Requirement already satisfied: huggingface-hub<1.0.0,>=0.14.0 in /usr/local/lib/python3.10/dist-packages (from datasets)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from datasets) (23.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from datasets) (6.0.1)
Requirement already satisfied: responses<0.19 in /usr/local/lib/python3.10/dist-packages (from evaluate) (0.18.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers) (3.12.4)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (2023.6.
Requirement already satisfied: tokenizers<0.15,>=0.14 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.
Requirement already satisfied: safetensors>=0.3.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.4.0)
Requirement already satisfied: absl-py in /usr/local/lib/python3.10/dist-packages (from rouge-score) (1.4.0)
Requirement already satisfied: six>=1.14.0 in /usr/local/lib/python3.10/dist-packages (from rouge-score) (1.16.0)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.3.2)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (23.1.
Requirement already satisfied: charset-normalizer<4.0,>=2.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->da
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in /usr/local/lib/python3.10/dist-packages (from aiohttp->dat
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.9.
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->datasets)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->dat
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->dat
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas->datasets)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->datasets) (2023.3.
```

If you're opening this notebook locally, make sure your environment has an install from the last version of those libraries.

To be able to share your model with the community and generate results like the one shown in the picture below via the inference API, there are a few more steps to follow.

First you have to store your authentication token from the Hugging Face website (sign up here if you haven't already!) then execute the following cell and input your username and password:

```
from huggingface_hub import notebook_login

notebook_login()
```

Token is valid (permission: write).

Your token has been saved in your configured git credential helpers (store).

Your token has been saved to /root/.cache/huggingface/token

Login successful

Then you need to install Git-LFS. Uncomment the following instructions:

```
# !apt install git-lfs
```

Make sure your version of Transformers is at least 4.11.0 since the functionality was introduced in that version:

```
import transformers

print(transformers.__version__)
```

```
4.34.0
```

You can find a script version of this notebook to fine-tune your model in a distributed fashion using multiple GPUs or TPUs here.

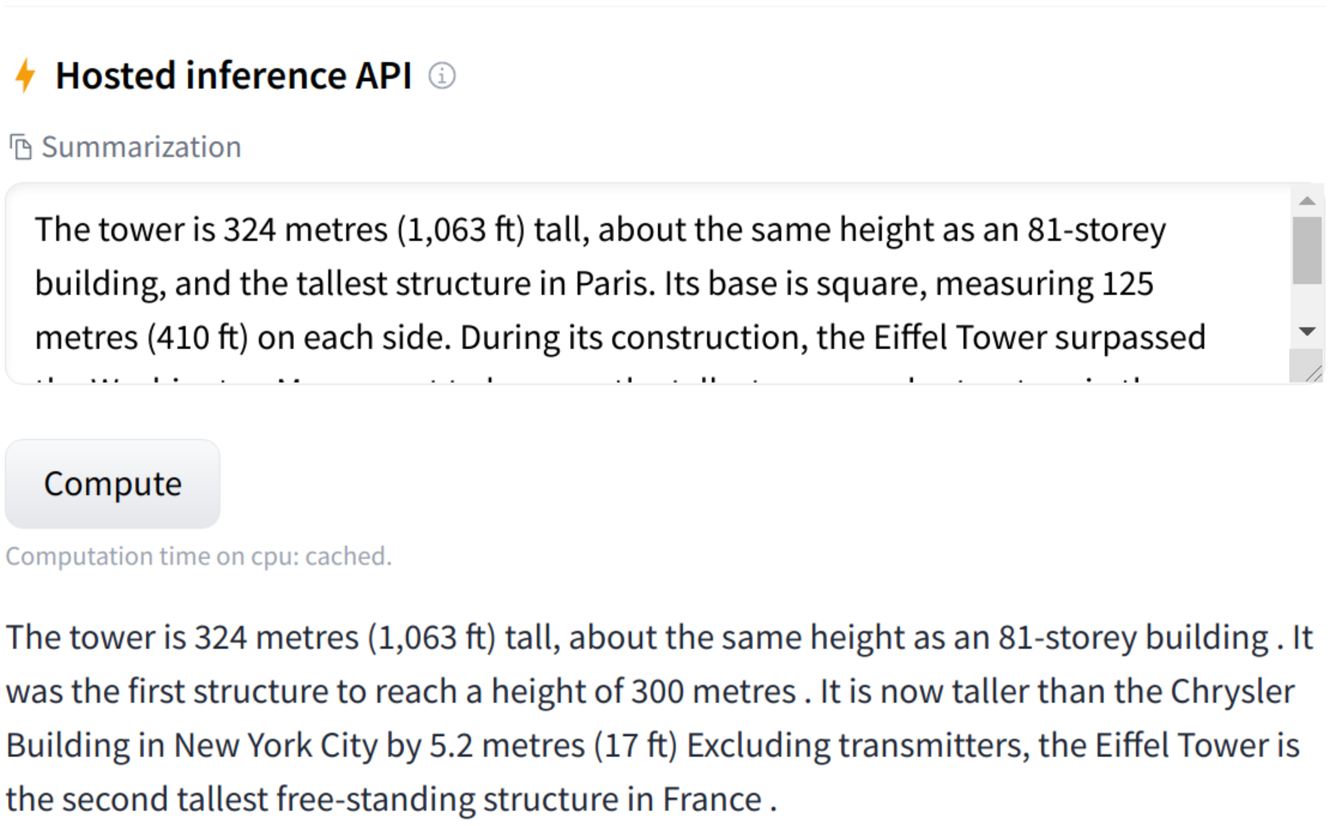
We also quickly upload some telemetry - this tells us which examples and software versions are getting used so we know where to prioritize our maintenance efforts. We don't collect (or care about) any personally identifiable information, but if you'd prefer not to be counted, feel free to skip this step or delete this cell entirely.

```
from transformers.utils import send_example_telemetry

send_example_telemetry("summarization_notebook", framework="pytorch")
```

Fine-tuning a model on a summarization task

In this notebook, we will see how to fine-tune one of the 🤗 Transformers model for a summarization task. We will use the XSum dataset (for extreme summarization) which contains BBC articles accompanied with single-sentence summaries.



The tower is 324 metres (1,063 ft) tall, about the same height as an 81-storey building, and the tallest structure in Paris. Its base is square, measuring 125 metres (410 ft) on each side. During its construction, the Eiffel Tower surpassed

Computation time on cpu: cached.

The tower is 324 metres (1,063 ft) tall, about the same height as an 81-storey building . It was the first structure to reach a height of 300 metres . It is now taller than the Chrysler Building in New York City by 5.2 metres (17 ft) Excluding transmitters, the Eiffel Tower is the second tallest free-standing structure in France .

We will see how to easily load the dataset for this task using 🤗 Datasets and how to fine-tune a model on it using the `Trainer` API.

```
model_checkpoint = "t5-small"
```

This notebook is built to run with any model checkpoint from the Model Hub as long as that model has a sequence-to-sequence version in the Transformers library. Here we picked the `t5-small` checkpoint.

Loading the dataset

We will use the 🤗 Datasets library to download the data and get the metric we need to use for evaluation (to compare our model to the benchmark). This can be easily done with the functions `load_dataset` and `load_metric`.

```
from datasets import load_dataset
from evaluate import load

raw_datasets = load_dataset("xsum")
metric = load("rouge")
```

The `dataset` object itself is [`DatasetDict`](), which contains one key for the training, validation and test set:

```
raw_datasets
```

```
    DatasetDict({
        train: Dataset({
            features: ['document', 'summary', 'id'],
            num_rows: 204045
        })
        validation: Dataset({
            features: ['document', 'summary', 'id'],
            num_rows: 11332
        })
        test: Dataset({
            features: ['document', 'summary', 'id'],
            num_rows: 11334
        })
    })
```

To access an actual element, you need to select a split first, then give an index:

```
raw_datasets["train"][0]
```

```
    {'document': 'The full cost of damage in Newton Stewart, one of the areas worst affected, is still being
    assessed.\nRepair work is ongoing in Hawick and many roads in Peeblesshire remain badly affected by standing
    water.\nTrains on the west coast mainline face disruption due to damage at the Lamington Viaduct.\nMany businesses and
    householders were affected by flooding in Newton Stewart after the River Cree overflowed into the town.\nFirst Minister
    Nicola Sturgeon visited the area to inspect the damage.\nThe waters breached a retaining wall, flooding many commercial
    properties on Victoria Street - the main shopping thoroughfare.\nJeanette Tate, who owns the Cinnamon Cafe which was
    badly affected, said she could not fault the multi-agency response once the flood hit.\nHowever, she said more
    preventative work could have been carried out to ensure the retaining wall did not fail.\n"It is difficult but I do
    think there is so much publicity for Dumfries and the Nith - and I totally appreciate that - but it is almost like
    we\'re neglected or forgotten," she said.\n"That may not be true but it is perhaps my perspective over the last few
    days.\n"Why were you not ready to help us a bit more when the warning and the alarm alerts had gone out?"\nMeanwhile, a
    flood alert remains in place across the Borders because of the constant rain.\nPeebles was badly hit by problems,
    sparking calls to introduce more defences in the area.\nScottish Borders Council has put a list on its website of the
    roads worst affected and drivers have been urged not to ignore closure signs.\nThe Labour Party\'s deputy Scottish
    leader Alex Rowley was in Hawick on Monday to see the situation first hand.\nHe said it was important to get the flood
    protection plan right but backed calls to speed up the process.\n"I was quite taken aback by the amount of damage that
    has been done," he said.\n"Obviously it is heart-breaking for people who have been forced out of their homes and the
    impact on businesses."\nHe said it was important that "immediate steps" were taken to protect the areas most vulnerable
    and a clear timetable put in place for flood prevention plans.\nHave you been affected by flooding in Dumfries and
    Galloway or the Borders? Tell us about your experience of the situation and how it was handled. Email us on
    selkirk.news@bbc.co.uk or dumfries@bbc.co.uk.',
     'summary': 'Clean-up operations are continuing across the Scottish Borders and Dumfries and Galloway after flooding
    caused by Storm Frank.',
     'id': '35232142'}
```

To get a sense of what the data looks like, the following function will show some examples picked randomly in the dataset.

```
import datasets
import random
import pandas as pd
from IPython.display import display, HTML

def show_random_elements(dataset, num_examples=5):
    assert num_examples <= len(dataset), "Can't pick more elements than there are in the dataset."
    picks = []
    for _ in range(num_examples):
        pick = random.randint(0, len(dataset)-1)
        while pick in picks:
            pick = random.randint(0, len(dataset)-1)
        picks.append(pick)

    df = pd.DataFrame(dataset[picks])
    for column, typ in dataset.features.items():
        if isinstance(typ, datasets.ClassLabel):
            df[column] = df[column].transform(lambda i: typ.names[i])
    display(HTML(df.to_html()))
```

```
show_random_elements(raw_datasets["train"])
```

	do
0	Gale, 30, was banned for two games after Yorkshire beat Lancashire in the County Championship on 3 September and must now face a disciplinary panel.\nT Telegraph reports Gale allegedly used the word 'Kolpak' to refer to South African Ashwell Prince.\n"We'll help him in any way possible," Moxon told the Y Post.\n"We are going to do what we can to help him clear his name."\nGale was banned for his team's final two County Championship matches of the seaso mandatory disciplinary procedure.\nThe England and Wales Cricket Board (ECB) has yet to confirm the nature of the hearing that is set to take place in the days, and has so far refused to comment on the disciplinary proceedings.\nIt is likely that the case against Gale will centre on whether his use of the word can be construed as being racist.\nIn Gale's absence, the White Rose county beat Nottinghamshire last Friday to win their 31st outright County Champ title.\nGale was informed on Thursday that he would not be allowed to lift the trophy with the team, but took to the field later on to join his team-mates.\nMea the county have confirmed that chairman Colin Graves, who is also deputy chairman of the ECB, will play no part in the disciplinary proceedings.\nGale's case handled by vice-chairman Robin Smith and chief executive Mark
1	Wycombe offered the 23-year-old a new deal in the summer and turned down a bid for him on deadline day in August.\n"I've been in exactly the same p Aaron)," said Ainsworth.\n"You feel loyal because this is the club that gave you the chance but you're also ambitious and have agents in your ears saying 'y sign a new contract as it's career suicide'."\nHe continued: "All I know is that when Aaron puts a shirt on he gives his absolute best for Wycombe. If he goes on a fantastic career in League One, the Championship or the Premier League, we have to be proud of him."\nSince moving to Adams Park from Brentford in 201 has made 127 appearances and Ainsworth has said he is "probably the best centre-half in League Two".\nHe told BBC Three Counties Radio: "I'm hoping he's May and June and still playing because that means nobody came in.\n"But I always live in fear because he's a fantastic player and I think he can play higher saying that because I know he might go o
2	Angelino Alfano's staunch opposition to surrogacy is part of a wider campaign by the minister against rights for gay and unmarried heterosexual couples.\nSu is "the most vile, illegal trade that man has invented," Alfano told Italy's Avvenire newspaper.\nItaly is the last major Western nation to deny same-sex couple on issues such as parenthood.\nIt was condemned last year by the European Court of Human Rights for failing to legislate on the issue.\nSurrogacy is c illegal in Italy, punishable by fines and up to two years in prison.\nMr Alfano's suggestion that the practice be classed in the same bracket as sex offence entail harsher punishments.\nHis small New Centre Right party (NCD) - a junior partner in the ruling coalition - opposes any form of surrogacy, claiming it s traditional family values.\n"Stepchild [adoption] really risks bringing the country closer to wombs-for-rent, towards the most vile, illegal trade that man has in he told the Roman Catholic newspaper.\n"We want wombs-for-rent to become a universal crime, which is punished with a jail term. Just as happens crimes."\nSome Italian couples have used surrogates abroad, but the status of their children is legally shaky and has led to prolonged battles in th courts.\nCentre-left Prime Minister Matteo Renzi had promised to legalise civil unions before the end of 2015 but coalition infighting has meant that no action been taken.\nThe contested bill is due to return to parliament on 26 January, with most of the tension focused on whether unmarried partners should be all adopt their stepchildren under certain circumstances.\nProponents of the plan say this would protect the rights of a child if its natural parent died. Opponer as Mr Alfano say it would open the way for gay couples to seek children via surrogate n
3	The 35-year-old is Wales' record cap holder with 121 but has been on the bench for Wales' first two Six Nations games, with Rob Evans preferred.\nCardi loose-head Jenkins admits the 2019 World Cup will be "too far" for him but is determined to regain his place in the Welsh front row.\n"I'm still as hungry as eve that Welsh jersey on," he said.\n"Even though I'm on the bench, I still think I've got something to offer.\n"It all depends on your regional form. I'd like to think if playing well week in, week out then I would have a chance of being involved.\n"I can't see me retiring because I'm still eligible to play for Wales."\nWales hea Warren Gatland sprang a major surprise when he selected Evans, 23, ahead of Jenkins for their Six Nations opener in Ireland.\nScarlet Evans kept his place Saturday's win against Scotland, with Jenkins appearing from the bench in both matches.\nAs a veteran of three British and Irish Lions tours, Jenkins is not being on the Welsh bench but has been impressed by Evans.\n"It's different for me but Rob's gone really well and I've enjoyed supporting him and trying to r impact off the bench," Jenkins added.\n"I think he's been outstanding in the first two games. He's still young at international level but learning every week.\ strong scrummager and carries very well around the field. He's only going to get better.\n"That's good for Wales that they've got someone coming through n will hopefully be there for the next 10 years.\n"It's quite hard watching, different nerves as to starting, when you know what to expect."\nAfter selecting Evans

The metric is an instance of `datasets.Metric`:

do that [retire] but at the moment I'm just concentrating on playing week in, week out for the region and playi

```
metric
```

```
EvaluationModule(name: "rouge", module_type: "metric", features: [{'predictions': Value(dtype='string', id='sequence'),
'references': Sequence(feature=Value(dtype='string', id='sequence'), length=-1, id=None)}, {'predictions':
Value(dtype='string', id='sequence'), 'references': Value(dtype='string', id='sequence')}], usage: """
Calculates average rouge scores for a list of hypotheses and references
Args:
    predictions: list of predictions to score. Each prediction
        should be a string with tokens separated by spaces.
    references: list of reference for each prediction. Each
        reference should be a string with tokens separated by spaces.
    rouge_types: A list of rouge types to calculate.
        Valid names:
        `"rouge{n}"` (e.g. `"rouge1"`, `"rouge2"`) where: {n} is the n-gram based scoring,
        `"rougeL"`: Longest common subsequence based scoring.
        `"rougeLsum"`: rougeLsum splits text using `"
"`.
        See details in https://github.com/huggingface/datasets/issues/617
    use_stemmer: Bool indicating whether Porter stemmer should be used to strip word suffixes.
    use_aggregator: Return aggregates if this is set to True
Returns:
    rouge1: rouge_1 (f1),
    rouge2: rouge_2 (f1),
    rougeL: rouge_l (f1),
    rougeLsum: rouge_lsum (f1)
Examples:

    >>> rouge = evaluate.load('rouge')
    >>> predictions = ["hello there", "general kenobi"]
    >>> references = ["hello there", "general kenobi"]
    >>> results = rouge.compute(predictions=predictions, references=references)
    >>> print(results)
    {'rouge1': 1.0, 'rouge2': 1.0, 'rougeL': 1.0, 'rougeLsum': 1.0}
""", stored examples: 0)
```

You can call its `compute` method with your predictions and labels, which need to be list of decoded strings:

```
fake_preds = ["hello there", "general kenobi"]
fake_labels = ["hello there", "general kenobi"]
```

```
metric.compute(predictions=fake_preds, references=fake_labels)
```

```
{'rouge1': 1.0, 'rouge2': 1.0, 'rougeL': 1.0, 'rougeLsum': 1.0}
```

Preprocessing the data

Before we can feed those texts to our model, we need to preprocess them. This is done by a 🤗 Transformers `Tokenizer` which will (as the name indicates) tokenize the inputs (including converting the tokens to their corresponding IDs in the pretrained vocabulary) and put it in a format the model expects, as well as generate the other inputs that the model requires.

To do all of this, we instantiate our tokenizer with the `AutoTokenizer.from_pretrained` method, which will ensure:

- we get a tokenizer that corresponds to the model architecture we want to use,
- we download the vocabulary used when pretraining this specific checkpoint.

That vocabulary will be cached, so it's not downloaded again the next time we run the cell.

```
from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained(model_checkpoint)
```

By default, the call above will use one of the fast tokenizers (backed by Rust) from the 🤗 Tokenizers library.

You can directly call this tokenizer on one sentence or a pair of sentences:

```
tokenizer("Hello, this one sentence!")
```

```
{'input_ids': [8774, 6, 48, 80, 7142, 55, 1], 'attention_mask': [1, 1, 1, 1, 1, 1, 1]}
```

Depending on the model you selected, you will see different keys in the dictionary returned by the cell above. They don't matter much for what we're doing here (just know they are required by the model we will instantiate later), you can learn more about them in this tutorial if you're interested.

Instead of one sentence, we can pass along a list of sentences:

```
tokenizer(["Hello, this one sentence!", "This is another sentence."])
```

```
{'input_ids': [[8774, 6, 48, 80, 7142, 55, 1], [100, 19, 430, 7142, 5, 1]], 'attention_mask': [[1, 1, 1, 1, 1, 1, 1],
[1, 1, 1, 1, 1, 1]]}
```

To prepare the targets for our model, we need to tokenize them using the `text_target` parameter. This will make sure the tokenizer uses the special tokens corresponding to the targets:

```
print(tokenizer(text_target=["Hello, this one sentence!", "This is another sentence."]))
```

```
{'input_ids': [[8774, 6, 48, 80, 7142, 55, 1], [100, 19, 430, 7142, 5, 1]], 'attention_mask': [[1, 1, 1, 1, 1, 1, 1], [1
```

If you are using one of the five T5 checkpoints we have to prefix the inputs with "summarize:" (the model can also translate and it needs the prefix to know which task it has to perform).

```
if model_checkpoint in ["t5-small", "t5-base", "t5-larg", "t5-3b", "t5-11b"]:
    prefix = "summarize: "
else:
    prefix = ""
```

We can then write the function that will preprocess our samples. We just feed them to the `tokenizer` with the argument `truncation=True`. This will ensure that an input longer that what the model selected can handle will be truncated to the maximum length accepted by the model. The padding will be dealt with later on (in a data collator) so we pad examples to the longest length in the batch and not the whole dataset.

```
max_input_length = 1024
max_target_length = 128

def preprocess_function(examples):
    inputs = [prefix + doc for doc in examples["document"]]
    model_inputs = tokenizer(inputs, max_length=max_input_length, truncation=True)
    # Setup the tokenizer for targets
    labels = tokenizer(text_target=examples["summary"], max_length=max_target_length, truncation=True)

    model_inputs["labels"] = labels["input_ids"]
    return model_inputs
```

This function works with one or several examples. In the case of several examples, the tokenizer will return a list of lists for each key:

```
preprocess_function(raw_datasets['train'][:2])
```

```
{'input_ids': [[21603, 10, 37, 423, 583, 13, 1783, 16, 20126, 16496, 6, 80, 13, 8, 844, 6025, 4161, 6, 19, 341, 271,
14841, 5, 7057, 161, 19, 4912, 16, 1626, 5981, 11, 186, 7540, 16, 1276, 15, 2296, 7, 5718, 2367, 14621, 4161, 57, 4125,
387, 5, 15059, 7, 30, 8, 4653, 4939, 711, 747, 522, 17879, 788, 12, 1783, 44, 8, 15763, 6029, 1813, 9, 7472, 5, 1404,
1623, 11, 5699, 277, 130, 4161, 57, 18368, 16, 20126, 16496, 227, 8, 2473, 5895, 15, 147, 89, 22411, 139, 8, 1511, 5,
1485, 3271, 3, 21926, 9, 472, 19623, 5251, 8, 616, 12, 15614, 8, 1783, 5, 37, 13818, 10564, 15, 26, 3, 9, 3, 19513,
1481, 6, 18368, 186, 1328, 2605, 30, 7488, 1887, 3, 18, 8, 711, 2309, 9517, 89, 355, 5, 3966, 1954, 9233, 15, 6, 113,
293, 7, 8, 16548, 13363, 106, 14022, 84, 47, 14621, 4161, 6, 243, 255, 228, 59, 7828, 8, 1249, 18, 545, 11298, 1773,
728, 8, 8347, 1560, 5, 611, 6, 255, 243, 72, 1709, 1528, 161, 228, 43, 118, 4006, 91, 12, 766, 8, 3, 19513, 1481, 410,
59, 5124, 5, 96, 196, 17, 19, 1256, 68, 27, 103, 317, 132, 19, 78, 231, 23546, 21, 970, 51, 89, 2593, 11, 8, 2504, 189,
3, 18, 11, 27, 3536, 3653, 24, 3, 18, 68, 34, 19, 966, 114, 62, 31, 60, 23708, 42, 11821, 976, 255, 243, 5, 96, 11880,
164, 59, 36, 1176, 68, 34, 19, 2361, 82, 3503, 147, 8, 336, 360, 477, 5, 96, 17891, 130, 25, 59, 1065, 12, 199, 178, 3,
9, 720, 72, 116, 8, 6337, 11, 8, 6196, 5685, 7, 141, 2767, 91, 4609, 7940, 6, 3, 9, 8347, 5685, 3048, 16, 286, 640, 8,
17600, 7, 250, 13, 8, 3917, 3412, 5, 1276, 15, 2296, 7, 47, 14621, 1560, 57, 982, 6, 13233, 53, 3088, 12, 4277, 72,
13613, 7, 16, 8, 616, 5, 12580, 17600, 7, 2063, 65, 474, 3, 9, 570, 30, 165, 475, 13, 8, 7540, 6025, 4161, 11, 3863,
43, 118, 3, 19492, 59, 12, 9751, 12493, 3957, 5, 37, 16117, 3450, 31, 7, 21108, 12580, 2488, 5104, 11768, 1306, 47, 16,
1626, 5981, 30, 2089, 12, 217, 8, 1419, 166, 609, 5, 216, 243, 34, 47, 359, 12, 129, 8, 8347, 1711, 515, 269, 68, 3,
9485, 3088, 12, 1634, 95, 8, 433, 5, 96, 196, 47, 882, 1026, 3, 9, 1549, 57, 8, 866, 13, 1783, 24, 65, 118, 612, 976,
3, 88, 243, 5, 96, 14116, 34, 19, 842, 18, 18087, 21, 151, 113, 43, 118, 5241, 91, 13, 70, 2503, 11, 8, 1113, 30, 1623,
535, 216, 243, 34, 47, 359, 24, 96, 603, 5700, 342, 2245, 121, 130, 1026, 12, 1822, 8, 844, 167, 9930, 11, 3, 9, 964,
97, 3869, 474, 16, 286, 21, 8347, 9793, 1390, 5, 2114, 25, 118, 4161, 57, 18368, 16, 970, 51, 89, 2593, 11, 10987, 32,
1343, 42, 8, 17600, 7, 58, 8779, 178, 81, 39, 351, 13, 8, 1419, 11, 149, 34, 47, 10298, 5, 8601, 178, 30, 142, 40, 157,
12546, 5, 15808, 1741, 115, 115, 75, 5, 509, 5, 1598, 42, 146, 51, 89, 2593, 1741, 115, 115, 75, 5, 509, 5, 1598, 5,
1], [21603, 10, 71, 1472, 6196, 877, 326, 44, 8, 9108, 86, 29, 16, 6000, 1887, 44, 81, 11484, 10, 1755, 272, 4209, 30,
1856, 11, 2554, 130, 1380, 12, 1175, 8, 1595, 5, 282, 79, 3, 9094, 1067, 79, 1509, 8, 192, 14264, 6, 3, 16669, 596, 18,
969, 18, 1583, 16, 8, 443, 2447, 6, 3, 35, 6106, 19565, 57, 12314, 7, 5, 555, 13, 8, 1552, 1637, 19, 45, 3434, 6, 8,
119, 45, 1473, 11, 14441, 5, 94, 47, 70, 166, 706, 16, 5961, 5316, 5, 37, 2535, 13, 80, 13, 8, 14264, 243, 186, 13, 8,
9234, 141, 646, 525, 12770, 7, 30, 1476, 11, 175, 141, 118, 10932, 5, 2867, 1637, 43, 13666, 3709, 11210, 11, 56, 1731,
70, 1552, 13, 8, 3457, 4939, 865, 145, 79, 141, 4355, 5, 5076, 43, 3958, 15, 26, 21, 251, 81, 8, 3211, 5, 86, 7, 102,
1955, 24723, 243, 10, 96, 196, 17, 3475, 38, 713, 8, 1472, 708, 365, 80, 13, 8, 14264, 274, 16436, 12, 8, 511, 5, 96,
27674, 8, 2883, 1137, 19, 341, 365, 4962, 6, 34, 19, 816, 24, 8, 1472, 47, 708, 24067, 535, 1]], 'attention_mask': [[1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]], 'labels': [[7433, 18, 413, 2673, 33, 6168, 640, 8, 12580, 17600, 7, 11,
970, 51, 89, 2593, 11, 10987, 32, 1343, 227, 18368, 2953, 57, 16133, 4937, 5, 1], [2759, 8548, 14264, 43, 118, 10932,
57, 1472, 16, 3, 9, 18024, 1584, 739, 3211, 16, 27874, 690, 2050, 5, 1]]}
```

To apply this function on all the pairs of sentences in our dataset, we just use the `map` method of our `dataset` object we created earlier. This will apply the function on all the elements of all the splits in `dataset`, so our training, validation and testing data will be preprocessed in one single command.

```
tokenized_datasets = raw_datasets.map(preprocess_function, batched=True)
```

Even better, the results are automatically cached by the 🤗 Datasets library to avoid spending time on this step the next time you run your notebook. The 🤗 Datasets library is normally smart enough to detect when the function you pass to map has changed (and thus requires to not use the cache data). For instance, it will properly detect if you change the task in the first cell and rerun the notebook. 🤗 Datasets warns you when it uses cached files, you can pass `load_from_cache_file=False` in the call to `map` to not use the cached files and force the preprocessing to be applied again.

Note that we passed `batched=True` to encode the texts by batches together. This is to leverage the full benefit of the fast tokenizer we loaded earlier, which will use multi-threading to treat the texts in a batch concurrently.

▾ Fine-tuning the model

Now that our data is ready, we can download the pretrained model and fine-tune it. Since our task is of the sequence-to-sequence kind, we use the `AutoModelForSeq2SeqLM` class. Like with the tokenizer, the `from_pretrained` method will download and cache the model for us.

```
import torch


from transformers import AutoModelForSeq2SeqLM, DataCollatorForSeq2Seq, Seq2SeqTrainingArguments, Seq2SeqTrainer
device = "cuda:0" if torch.cuda.is_available() else "cpu"
model = AutoModelForSeq2SeqLM.from_pretrained(model_checkpoint)
model = model.to(device)
```

Note that we don't get a warning like in our classification example. This means we used all the weights of the pretrained model and there is no randomly initialized head in this case.

To instantiate a `Seq2SeqTrainer`, we will need to define three more things. The most important is the [`Seq2SeqTrainingArguments`](), which is a class that contains all the attributes to customize the training. It requires one folder name, which will be used to save the checkpoints of the model, and all other arguments are optional:

```
!pip install transformers[torch]
```

```
--NORMAL--

Requirement already satisfied: transformers[torch] in /usr/local/lib/python3.10/dist-packages (4.34.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers[torch]) (3.12.4)
Requirement already satisfied: huggingface-hub<1.0,>=0.16.4 in /usr/local/lib/python3.10/dist-packages (from transformer
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from transformers[torch]) (1.23.
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from transformers[torch]) (2
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformers[torch]) (6.0.1
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers[torch])
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from transformers[torch]) (2.31.0)
Requirement already satisfied: tokenizers<0.15,>=0.14 in /usr/local/lib/python3.10/dist-packages (from transformers[tor
Requirement already satisfied: safetensors>=0.3.1 in /usr/local/lib/python3.10/dist-packages (from transformers[torch])
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packages (from transformers[torch]) (4.66.1
Requirement already satisfied: torch!=1.12.0,>=1.10 in /usr/local/lib/python3.10/dist-packages (from transformers[torch
Requirement already satisfied: accelerate>=0.20.3 in /usr/local/lib/python3.10/dist-packages (from transformers[torch])
Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages (from accelerate>=0.20.3->transformers
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.16.4->tr
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch!=1.12.0,>=1.10->transformer
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch!=1.12.0,>=1.10->transfor
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch!=1.12.0,>=1.10->transforme
Requirement already satisfied: triton==2.0.0 in /usr/local/lib/python3.10/dist-packages (from torch!=1.12.0,>=1.10->tra
Requirement already satisfied: cmake in /usr/local/lib/python3.10/dist-packages (from triton==2.0.0->torch!=1.12.0,>=1.
Requirement already satisfied: lit in /usr/local/lib/python3.10/dist-packages (from triton==2.0.0->torch!=1.12.0,>=1.10
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->tran
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->transformers[tor
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->transforme
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->transforme
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch!=1.12.0,>
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-packages (from sympy->torch!=1.12.0,>=1.1
```

```
batch_size = 16
model_name = model_checkpoint.split("/")[-1]
args = Seq2SeqTrainingArguments(
    f"{model_name}-finetuned-xsum",
    eval_steps=200,
    evaluation_strategy = "steps",
    learning_rate=2e-5,
    per_device_train_batch_size=batch_size,
    per_device_eval_batch_size=batch_size,
    weight_decay=0.01,
    save_total_limit=3,
    num_train_epochs=10,
    predict_with_generate=True,
    fp16=True,
    push_to_hub=True,
)
```

Here we set the evaluation to be done at the end of each epoch, tweak the learning rate, use the `batch_size` defined at the top of the cell and customize the weight decay. Since the `Seq2SeqTrainer` will save the model regularly and our dataset is quite large, we tell it to make three saves maximum. Lastly, we use the `predict_with_generate` option (to properly generate summaries) and activate mixed precision training (to go a bit faster).

The last argument to setup everything so we can push the model to the [Hub]() regularly during training. Remove it if you didn't follow the installation steps at the top of the notebook. If you want to save your model locally in a name that is different than the name of the repository it will be pushed, or if you want to push your model under an organization and not your name space, use the `hub_model_id` argument to set the repo name (it needs to be the full name, including your namespace: for instance `"sgugger/t5-finetuned-xsum"` or `"huggingface/t5-finetuned-xsum"`).

Then, we need a special kind of data collator, which will not only pad the inputs to the maximum length in the batch, but also the labels:

```
data_collator = DataCollatorForSeq2Seq(tokenizer, model=model)
```

The last thing to define for our `Seq2SeqTrainer` is how to compute the metrics from the predictions. We need to define a function for this, which will just use the `metric` we loaded earlier, and we have to do a bit of pre-processing to decode the predictions into texts:

```
import nltk
import numpy as np

def compute_metrics(eval_pred):
    predictions, labels = eval_pred
    decoded_preds = tokenizer.batch_decode(predictions, skip_special_tokens=True)
    # Replace -100 in the labels as we can't decode them.
    labels = np.where(labels != -100, labels, tokenizer.pad_token_id)
    decoded_labels = tokenizer.batch_decode(labels, skip_special_tokens=True)

    # Rouge expects a newline after each sentence
    decoded_preds = ["\n".join(nltk.sent_tokenize(pred.strip())) for pred in decoded_preds]
    decoded_labels = ["\n".join(nltk.sent_tokenize(label.strip())) for label in decoded_labels]

    # Note that other metrics may not have a `use_aggregator` parameter
    # and thus will return a list, computing a metric for each sentence.
    result = metric.compute(predictions=decoded_preds, references=decoded_labels, use_stemmer=True, use_aggregator=True)
    # Extract a few results
    result = {key: value * 100 for key, value in result.items()}

    # Add mean generated length
    prediction_lens = [np.count_nonzero(pred != tokenizer.pad_token_id) for pred in predictions]
    result["gen_len"] = np.mean(prediction_lens)

    return {k: round(v, 4) for k, v in result.items()}
```

Then we just need to pass all of this along with our datasets to the `Seq2SeqTrainer`:

```
 tokenized_datasets["train"] .select(range(2000))
```

```
Dataset({
    features: ['document', 'summary', 'id', 'input_ids', 'attention_mask', 'labels'],
    num_rows: 2000
})
```

```
trainer = Seq2SeqTrainer(
    model,
    args,
    train_dataset=tokenized_datasets["train"].select(range(20000)),
    eval_dataset=tokenized_datasets["validation"].select(range(1000)),
    data_collator=data_collator,
    tokenizer=tokenizer,
    compute_metrics=compute_metrics
)
```

We can now finetune our model by just calling the `train` method:

```
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

```
trainer.train()
```

•••

[3801/12500 1:08:52 < 2:37:42, 0.92 it/s, Epoch 3.04/10]

Step	Training Loss	Validation Loss	Rouge1	Rouge2	Rougel	Rougelsum	Gen Len
200	No log	2.554310	27.398900	7.161800	21.458300	21.468100	18.827000
400	No log	2.546865	27.681700	7.276200	21.761100	21.779500	18.829000
600	2.818500	2.540582	27.617300	7.174100	21.678700	21.683200	18.770000
800	2.818500	2.537120	27.734500	7.402500	21.766000	21.772200	18.821000
1000	2.785800	2.530087	27.542300	7.282800	21.746200	21.725800	18.812000
1200	2.785800	2.527280	27.880600	7.481000	22.049300	22.044000	18.796000
1400	2.785800	2.525676	27.880800	7.609700	22.147200	22.138700	18.804000
1600	2.755700	2.520899	28.028800	7.605500	22.113300	22.102600	18.787000
1800	2.755700	2.514463	28.040000	7.630800	22.277300	22.266500	18.831000
2000	2.755600	2.511996	28.063900	7.620000	22.222300	22.198200	18.829000
2200	2.755600	2.508956	28.289900	7.698400	22.263200	22.258800	18.835000
2400	2.755600	2.505203	28.250700	7.610900	22.408600	22.399300	18.833000
2600	2.747700	2.504373	28.306800	7.788500	22.451600	22.442400	18.842000
2800	2.747700	2.501968	28.238000	7.771300	22.244400	22.211300	18.833000
3000	2.722900	2.499167	28.457300	7.633900	22.367200	22.354500	18.835000
3200	2.722900	2.496926	28.466800	7.814900	22.499800	22.461400	18.835000
3400	2.722900	2.494320	28.458500	7.675800	22.510900	22.482000	18.827000
3600	2.719900	2.492529	28.240000	7.635800	22.253800	22.241200	18.825000

[41/63 00:37 < 00:20, 1.08 it/s]

```
/usr/local/lib/python3.10/dist-packages/transformers/generation/utils.py:1260
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/transformers/generation/utils.py:1260
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/transformers/generation/utils.py:1260
  warnings.warn(
```

You can now upload the result of the training to the Hub, just execute this instruction:

```
/usr/local/lib/python3.10/dist-packages/transformers/generation/utils.py:1260
```

```
# trainer.push_to_hub()
  warnings.warn(
```

You can now share this model with all your friends, family, favorite pets: they can all load it with the identifier `"your-username/the-name-you-picked"` so for instance:

```
from transformers import AutoModelForSeq2SeqLM

model = AutoModelForSeq2SeqLM.from_pretrained("sgugger/my-awesome-model")
```