HuggingFace (HF)

Rowel Atienza, PhD

University of the Philippines

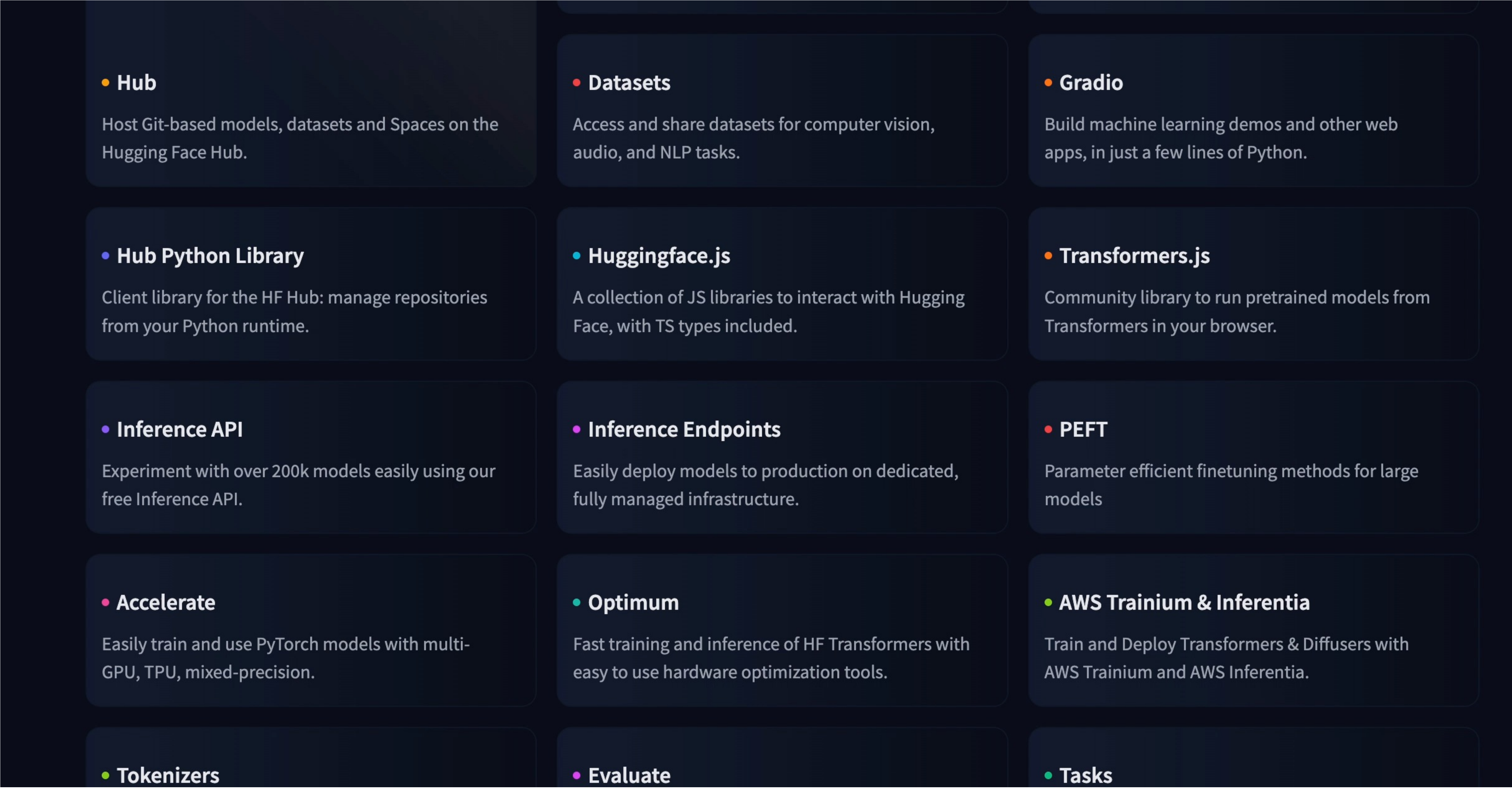
github.com/roatienza

2023

Why HuggingFace

- Task, Experience, Performance :
  - (Transformers, timm, Diffusers), Datasets, Evaluate
- More
  - Accelerate, Gradio, PEFT, Inference Endpoints, Solutions, etc

https://huggingface.co/docs



Hub

Host Git-based models, datasets and Spaces on the Hugging Face Hub.

Datasets

Access and share datasets for computer vision, audio, and NLP tasks.

Gradio

Build machine learning demos and other web apps, in just a few lines of Python.

Hub Python Library

Client library for the HF Hub: manage repositories from your Python runtime.

Huggingface.js

A collection of JS libraries to interact with Hugging Face, with TS types included.

Transformers.js

Community library to run pretrained models from Transformers in your browser.

Inference API

Experiment with over 200k models easily using our free Inference API.

Inference Endpoints

Easily deploy models to production on dedicated, fully managed infrastructure.

PEFT

Parameter efficient finetuning methods for large models

Accelerate

Easily train and use PyTorch models with multi-GPU, TPU, mixed-precision.

Optimum

Fast training and inference of HF Transformers with easy to use hardware optimization tools.

AWS Trainium & Inferentia

Train and Deploy Transformers & Diffusers with AWS Trainium and AWS Inferentia.

Tokenizers

Evaluate

Tasks

HuggingFace

Datasets

Datasets

- Datasets is a library for easily accessing and sharing datasets for Audio, Computer Vision, and Natural Language Processing (NLP) tasks.

```
pip install datasets
```

Load a dataset builder and inspect a dataset’s attributes without committing to downloading it

```
from datasets import load_dataset_builder
ds_builder = load_dataset_builder("wikitext",
"wikitext-2-raw-v1")
ds_builder.info.description
' The WikiText language modeling dataset is a
collection of over 100 million tokens extracted
from the set of verified\n Good and Featured
articles on Wikipedia. The dataset is available
under the Creative Commons Attribution-
ShareAlike\n License.\n'
```

HuggingFace

Usage

```
ds_builder.info.features
{'text': Value(dtype='string', id=None)}
```

If you're happy with the dataset, then load it with `load_dataset()`

```
from datasets import load_dataset
```

```
raw_dataset = load_dataset("wikitext",
"wikitext-2-raw-v1")
```

Sample datapoint

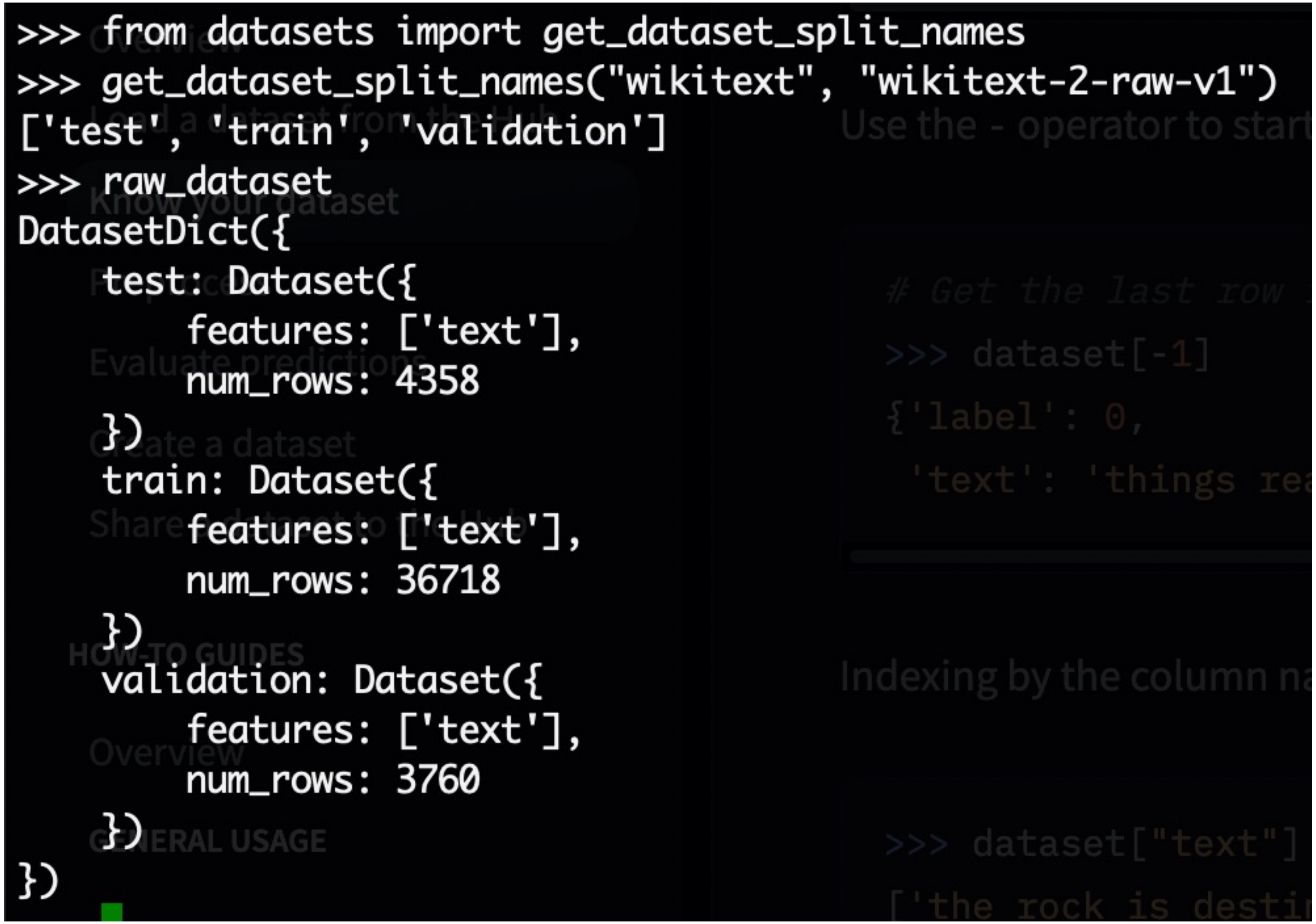
```
print(raw_dataset['train'][4])
{'text': " The game began development in 2010 ,
carrying over a large portion of the work done on
Valkyria Chronicles II . While it retained the
standard features of the series , it also underwent
multiple adjustments , such as making the game more
forgiving for series newcomers . Character designer
Raita Honjou and composer Hitoshi Sakimoto both
returned from previous entries , along with Valkyria
Chronicles II director Takeshi Ozawa . A large team
of writers handled the script . The game 's opening
theme was sung by May 'n . \n"}
```

HuggingFace

Split names

```
from datasets import get_dataset_split_names
get_dataset_split_names("wikitext", "wikitext-2-raw-v1")
['test', 'train', 'validation']
```

Split names



```
>>> from datasets import get_dataset_split_names
>>> get_dataset_split_names("wikitext", "wikitext-2-raw-v1")
['test', 'train', 'validation']
>>> raw_dataset
DatasetDict({
    test: Dataset({
        features: ['text'],
        num_rows: 4358
    })
    train: Dataset({
        features: ['text'],
        num_rows: 36718
    })
    validation: Dataset({
        features: ['text'],
        num_rows: 3760
    })
})
```

IterableDataset

Allows you to access and use the dataset without waiting for it to download completely

Note that unlike `Dataset`, data access is not random

`IterableDataset,` set `Dataset` to `streaming=True`

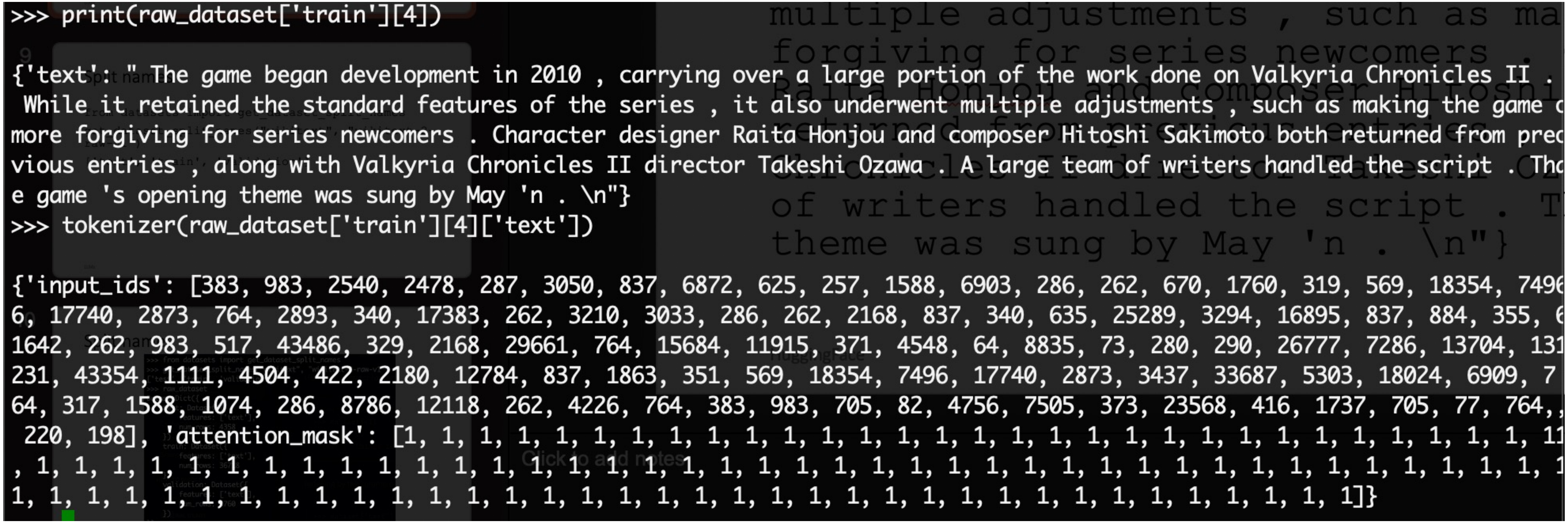
```
iterable_dataset = load_dataset("wikitext",
     "wikitext-2-raw-v1", streaming=True)
```

Preprocessing (e.g. Tokenization)

```
from transformers import GPT2Tokenizer
tokenizer =
    GPT2Tokenizer.from_pretrained('gpt2')
```

HuggingFace

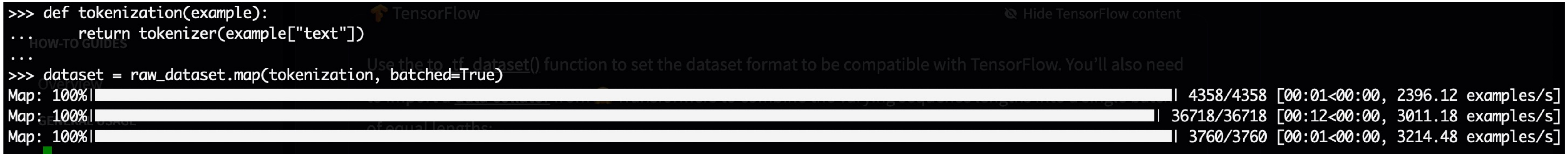
Preprocessing



```
>>> print(raw_dataset['train'][4])

{'text': " The game began development in 2010 , carrying over a large portion of the work done on Valkyria Chronicles II .
 While it retained the standard features of the series , it also underwent multiple adjustments , such as making the game c
more forgiving for series newcomers . Character designer Raita Honjou and composer Hitoshi Sakimoto both returned from prec
vious entries , along with Valkyria Chronicles II director Takeshi Ozawa . A large team of writers handled the script . Th
e game 's opening theme was sung by May 'n . \n"}
>>> tokenizer(raw_dataset['train'][4]['text'])

{'input_ids': [383, 983, 2540, 2478, 287, 3050, 837, 6872, 625, 257, 1588, 6903, 286, 262, 670, 1760, 319, 569, 18354, 7496
6, 17740, 2873, 764, 2893, 340, 17383, 262, 3210, 3033, 286, 262, 2168, 837, 340, 635, 25289, 3294, 16895, 837, 884, 355, 6
1642, 262, 983, 517, 43486, 329, 2168, 29661, 764, 15684, 11915, 371, 4548, 64, 8835, 73, 280, 290, 26777, 7286, 13704, 131
231, 43354, 1111, 4504, 422, 2180, 12784, 837, 1863, 351, 569, 18354, 7496, 17740, 2873, 3437, 33687, 5303, 18024, 6909, 7
64, 317, 1588, 1074, 286, 8786, 12118, 262, 4226, 764, 383, 983, 705, 82, 4756, 7505, 373, 23568, 416, 1737, 705, 77, 764,,
 220, 198], '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, 11
, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}
```

Fast Tokenization using `map`

```
def tokenization(example):
    return tokenizer(example["text"])

dataset = raw_dataset.map(tokenization,
    batched=True)
```



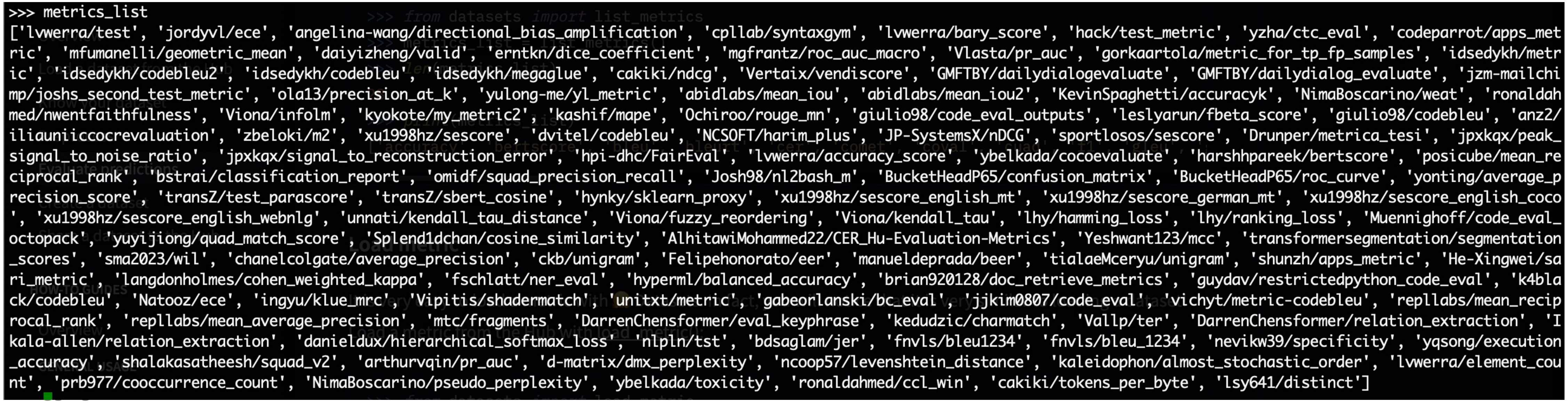
```
>>> def tokenization(example):
...     return tokenizer(example["text"])
...
>>> dataset = raw_dataset.map(tokenization, batched=True)
Map: 100%|████████████████████████████████████████| 4358/4358 [00:01<00:00, 2396.12 examples/s]
Map: 100%|███████████████████████████████████████| 36718/36718 [00:12<00:00, 3011.18 examples/s]
Map: 100%|████████████████████████████████████████| 3760/3760 [00:01<00:00, 3214.48 examples/s]
```

HuggingFace

Metrics

- `evaluate` provides various common and NLP-specific metrics for you to measure your models performance.

```
import evaluate

metrics_list = evaluate.list_evaluation_modules()
```



```
>>> metrics_list
['lvwerra/test', 'jordyvl/ece', 'angelina-wang/directional_bias_amplification', 'cpllab/syntaxgym', 'lvwerra/bary_score', 'hack/test_metric', 'yzha/ctc_eval', 'codeparrot/apps_met
ric', 'mfumanelli/geometric_mean', 'daiyizheng/valid', 'erntkn/dice_coefficient', 'mgfrantz/roc_auc_macro', 'Vlasta/pr_auc', 'gorkaartola/metric_for_tp_fp_samples', 'idsedykh/metr
ic', 'idsedykh/codebleu2', 'idsedykh/codebleu', 'idsedykh/megaglue', 'cakiki/ndcg', 'Vertaix/vendiscore', 'GMFTBY/dailydialogevaluate', 'GMFTBY/dailydialog_evaluate', 'jzm-mailchi
mp/joshs_second_test_metric', 'ola13/precision_at_k', 'yulong-me/yl_metric', 'abidlabs/mean_iou', 'abidlabs/mean_iou2', 'KevinSpaghetti/accuracyk', 'NimaBoscarino/weat', 'ronaldah
med/nwentfaithfulness', 'Viona/infolm', 'kyokote/my_metric2', 'kashif/mape', 'Ochiroo/rouge_mn', 'giulio98/code_eval_outputs', 'leslyarun/fbeta_score', 'giulio98/codebleu', 'anz2/
iliauniiccocrevaluation', 'zbeloki/m2', 'xu1998hz/sescore', 'dvitel/codebleu', 'NCSOFT/harim_plus', 'JP-SystemsX/nDCG', 'sportlosos/sescore', 'Drunper/metrica_tesi', 'jpxkqx/peak_
signal_to_noise_ratio', 'jpxkqx/signal_to_reconstruction_error', 'hpi-dhc/FairEval', 'lvwerra/accuracy_score', 'ybelkada/cocoevaluate', 'harshhpareek/bertscore', 'posicube/mean_re
ciprocal_rank', 'bstrai/classification_report', 'omidf/squad_precision_recall', 'Josh98/nl2bash_m', 'BucketHeadP65/confusion_matrix', 'BucketHeadP65/roc_curve', 'yonting/average_p
recision_score', 'transZ/test_parascore', 'transZ/sbert_cosine', 'hynky/sklearn_proxy', 'xu1998hz/sescore_english_mt', 'xu1998hz/sescore_german_mt', 'xu1998hz/sescore_english_coco
', 'xu1998hz/sescore_english_webnlg', 'unnati/kendall_tau_distance', 'Viona/fuzzy_reordering', 'Viona/kendall_tau', 'lhy/hamming_loss', 'lhy/ranking_loss', 'Muennighoff/code_eval_
octopack', 'yuyijiong/quad_match_score', 'Splend1dchan/cosine_similarity', 'AlhitawiMohammed22/CER_Hu-Evaluation-Metrics', 'Yeshwant123/mcc', 'transformersegmentation/segmentation
_scores', 'sma2023/wil', 'chanelcolgate/average_precision', 'ckb/unigram', 'Felipehonorato/eer', 'manueldeprada/beer', 'tialaeMceryu/unigram', 'shunzh/apps_metric', 'He-Xingwei/sa
ri_metric', 'langdonholmes/cohen_weighted_kappa', 'fschlatt/ner_eval', 'hyperml/balanced_accuracy', 'brian920128/doc_retrieve_metrics', 'guydav/restrictedpython_code_eval', 'k4bla
ck/codebleu', 'Natooz/ece', 'ingyu/klue_mrc', 'Vipitis/shadermatch', 'unitxt/metric', 'gabeorlanski/bc_eval', 'jjkim0807/code_eval', 'vichyt/metric-codebleu', 'repllabs/mean_recip
rocal_rank', 'repllabs/mean_average_precision', 'mtc/fragments', 'DarrenChensformer/eval_keyphrase', 'kedudzic/charmatch', 'Vallp/ter', 'DarrenChensformer/relation_extraction', 'I
kala-allen/relation_extraction', 'danieldux/hierarchical_softmax_loss', 'nlpln/tst', 'bdsaglam/jer', 'fnvls/bleu1234', 'fnvls/bleu_1234', 'nevikw39/specificity', 'yqsong/execution
_accuracy', 'shalakasatheesh/squad_v2', 'arthurvqin/pr_auc', 'd-matrix/dmx_perplexity', 'ncoop57/levenshtein_distance', 'kaleidophon/almost_stochastic_order', 'lvwerra/element_cou
nt', 'prb977/cooccurrence_count', 'NimaBoscarino/pseudo_perplexity', 'ybelkada/toxicity', 'ronaldahmed/ccl_win', 'cakiki/tokens_per_byte', 'lsy641/distinct']
```

Metrics

```
from datasets import list_metrics
metrics_list = list_metrics()
```

```
['accuracy', 'bertscore', 'bleu', 'bleurt',
'brier_score', 'cer', 'character', 'charcut_mt', 'chrf',
'code_eval', 'comet', 'competition_math', 'coval',
'cuad', 'exact_match', 'f1', 'frugalscore', 'glue',
'google_bleu', 'indic_glue', 'mae', 'mahalanobis',
'mape', 'mase', 'matthews_correlation', 'mauve',
'mean_iou', 'meteor', 'mse', 'nist_mt', 'pearsonr',
'perplexity', 'poseval', 'precision', 'r_squared',
'recall', 'rl_reliability', 'roc_auc', 'rouge',
'sacrebleu', 'sari', 'seqeval', 'smape', 'spearmanr',
'squad', 'squad_v2', 'super_glue', 'ter', 'trec_eval',
'wer', 'wiki_split', 'xnli', 'xtreme_s’,…
```

HuggingFace

Transformers

Transformers

- Natural Language Processing
- Computer Vision
- Audio
- Multi-modal

Transformers Models Implemented

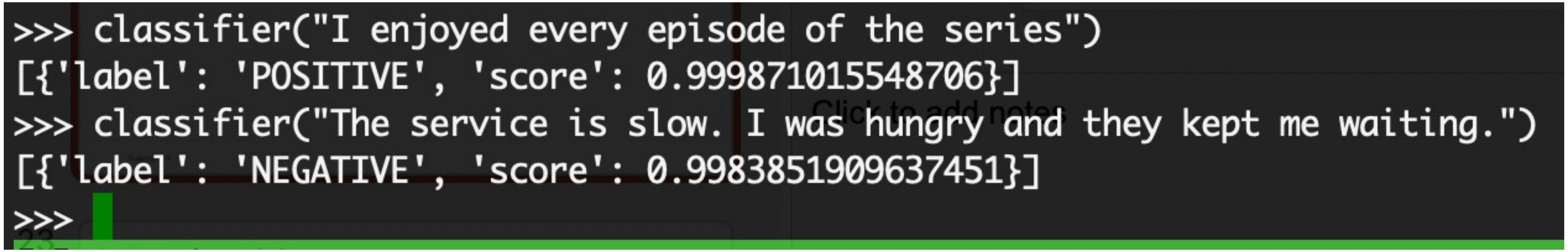
- Natural Language Processing – LLaMA2, BERT, GPTs, Mistral, Mixtral, etc
- Computer Vision – YOLOS, ViTDet, MobileViT, etc
- Audio – Whisper, VITS, etc
- Multi-modal – LLaVA, etc

Installation

```
pip install transformers datasets
```

Quick Use

```
from transformers import pipeline
classifier = pipeline("sentiment-analysis")
```

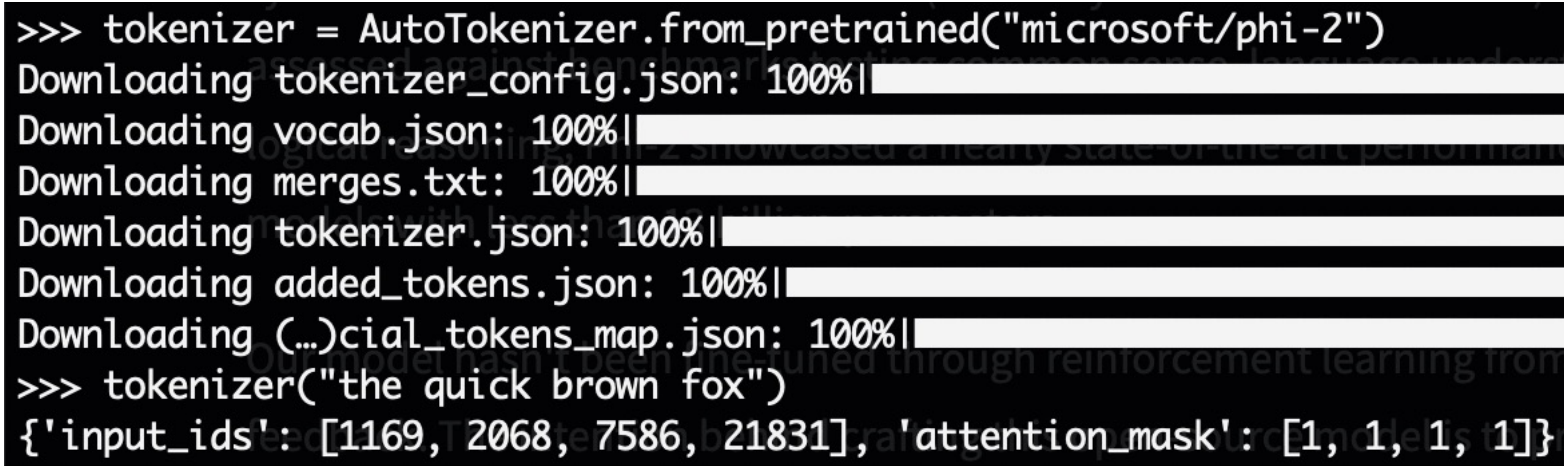


```
>>> classifier("I enjoyed every episode of the series")
[{'label': 'POSITIVE', 'score': 0.999871015548706}]
>>> classifier("The service is slow. I was hungry and they kept me waiting.")
[{'label': 'NEGATIVE', 'score': 0.9983851909637451}]
>>>
```

HuggingFace

Tokenizer

```
from transformers import AutoTokenizer
```



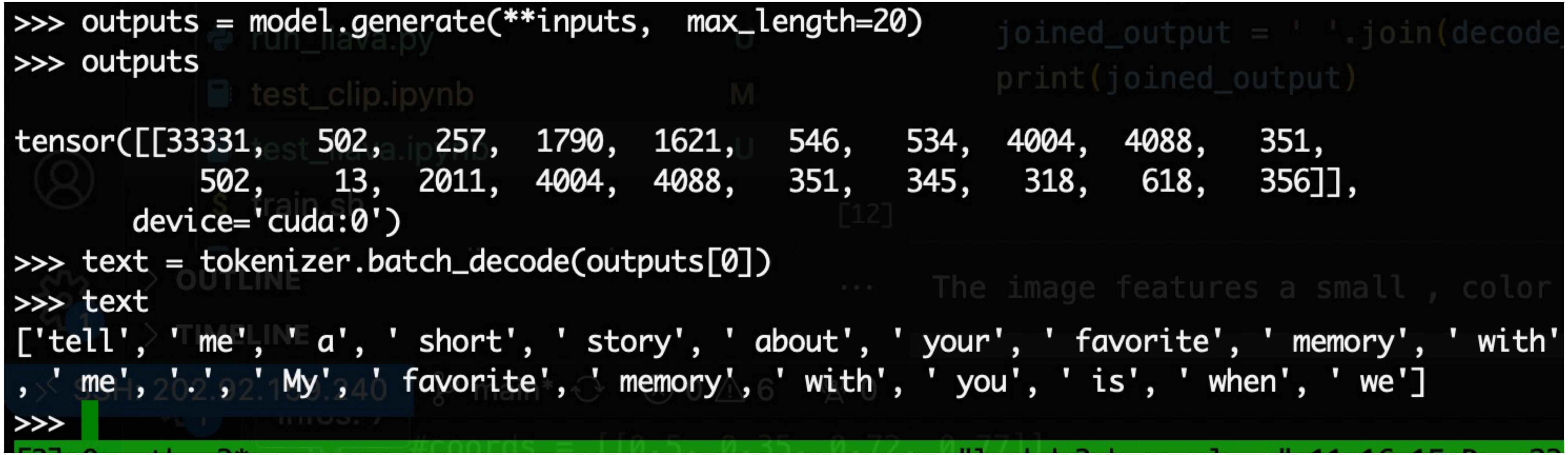
```
>>> tokenizer = AutoTokenizer.from_pretrained("microsoft/phi-2")
Downloading tokenizer_config.json: 100%|
Downloading vocab.json: 100%|
Downloading merges.txt: 100%|
Downloading tokenizer.json: 100%|
Downloading added_tokens.json: 100%|
Downloading (…)cial_tokens_map.json: 100%|
>>> tokenizer("the quick brown fox")
{'input_ids': [1169, 2068, 7586, 21831], 'attention_mask': [1, 1, 1, 1]}
```

HuggingFace

Model

```
from transformers import AutoModelForCausalLM
model = AutoModelForCausalLM.from_pretrained("microsoft/phi-2").cuda()
tokenizer = AutoTokenizer.from_pretrained("microsoft/phi-2")
inputs = tokenizer("tell me a short story", return_tensors="pt",
              return_attention_mask=False))
inputs['input_ids'] = inputs['input_ids'].cuda()
outputs = model.generate(**inputs,  max_length=20)
```

Model



```
>>> outputs = model.generate(**inputs,  max_length=20)
>>> outputs

tensor([[33331,   502,   257,  1790,  1621,   546,   534,  4004,  4088,   351,
           502,    13,  2011,  4004,  4088,   351,   345,   318,   618,   356]],
       device='cuda:0')
>>> text = tokenizer.batch_decode(outputs[0])
>>> text
['tell', ' me', ' a', ' short', ' story', ' about', ' your', ' favorite', ' memory', ' with'
, ' me', '.', ' My', ' favorite', ' memory', ' with', ' you', ' is', ' when', ' we']
>>>
```

run_llava.py
test_clip.ipynb
M
joined_output = ' '.join(decode
print(joined_output)
OUTLINE
TIMELINE
The image features a small , color

HuggingFace

Model Training

- `Trainer` - class optimized for training 🤗 Transformers models, making it easier to start training without manually writing your own training loop.

Steps

- Dataset
- Model and Tokenizer
- Trainer

Trainer Arguments

TrainingArguments class which contains all the hyperparameters you can tune as well as flags for activating different training options.

```
from transformers import TrainingArguments
training_args =
    TrainingArguments(output_dir="test_trainer")
```

Evaluate

- Trainer does not automatically evaluate model performance during training.
- Pass the `Trainer` a function to compute and report metrics.
- The 🤗 Evaluate library provides a simple accuracy function you can load with the `evaluate.load()`

Evaluate

```
import evaluate
metric = evaluate.load("accuracy")
```

Evaluate

- Post-processing functions are needed before using raw model predictions

```
def compute_metrics(eval_pred):
    logits, labels = eval_pred
    predictions = np.argmax(logits, axis=-1)
    return metric.compute(predictions=predictions,
     references=labels)
```

Trainer – start the training!

```
from transformers import Trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=eval_dataset,
    compute_metrics=compute_metrics,
)
trainer.train()
```

Try Examples

https://github.com/huggingface/transformers/tree/main/examples/pytorch/language-modeling

End