Hugging Face model navigation

MULTI-MODAL MODELS WITH HUGGING FACE

James Chapman
Curriculum Manager, DataCamp

Meet your instructor...

Sean Benson

- Former **particle physicist** at the Large Hadron Collider (LHC) in *CERN*, Switzerland
- Now applied **AI researcher** at the *Amsterdam University Medical Center*

Modalities covered

Text → Here are five ways to learn about AI...

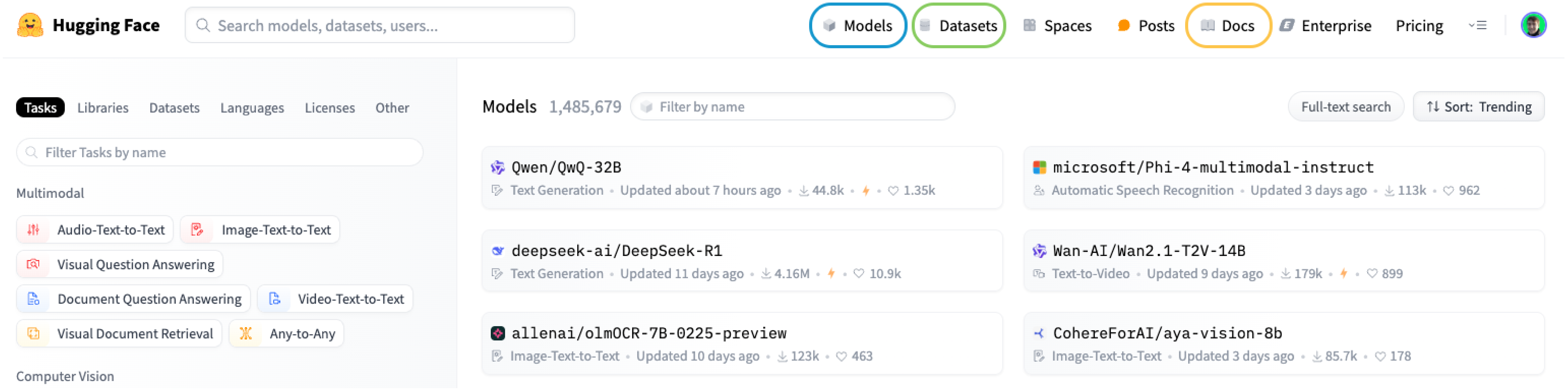
Image →

Audio →

Video →

[1] Image Credits: OpenAI generated

Hugging Face


Hugging Face | Search models, datasets, users... | Models | Datasets | Spaces | Posts | Docs | Enterprise | Pricing

Tasks | Libraries | Datasets | Languages | Licenses | Other

Filter Tasks by name

Multimodal

Audio-Text-to-Text | Image-Text-to-Text | Visual Question Answering | Document Question Answering | Video-Text-to-Text | Visual Document Retrieval | Any-to-Any

Computer Vision

Models 1,485,679 | Filter by name | Full-text search | Sort: Trending

Qwen/QwQ-32B
Text Generation • Updated about 7 hours ago • 44.8k • • 1.35k

microsoft/Phi-4-multimodal-instruct
Automatic Speech Recognition • Updated 3 days ago • 113k • 962

deepseek-ai/DeepSeek-R1
Text Generation • Updated 11 days ago • 4.16M • • 10.9k

Wan-AI/Wan2.1-T2V-14B
Text-to-Video • Updated 9 days ago • 179k • • 899

allenai/olmOCR-7B-0225-preview
Image-Text-to-Text • Updated 10 days ago • 123k • 463

CohereForAI/aya-vision-8b
Image-Text-to-Text • Updated 3 days ago • 85.7k • 178

- 1M+ models
- 250k+ datasets

The Hub API

- Access models and datasets programmatically

```
pip install huggingface_hub[cli]
```

Log in to access the models in your account:

```
>>> huggingface-cli login
```

- **All libraries, models, and datasets will be pre-installed**

Searching for models

- **Hugging Face API:** programmatic access to models and datasets

```
from huggingface_hub import HfApi
api = HfApi()
models = api.list_models()
```

- `task` : `"image-classification"` , `"text-to-image"` , etc.
- `sort` : e.g., `"likes"` or `"downloads"`
- `limit` : Maximum entries
- `tags` : Associated extra info of the model

Searching for models

```python
models = api.list_models(
    task="text-to-image",
    author="CompVis",
    tags="diffusers:StableDiffusionPipeline",
    sort="downloads"
)
```

```python
top_model = list(models)[0]
print(top_model)
```

```
ModelInfo(id='CompVis/stable-diffusion-v1-4', private=False, downloads=1097285,
          likes=6718, library_name='diffusers', ...
```

Using models from the API

```python
top_model_id = top_model.id
print(top_model_id)
```

```
CompVis/stable-diffusion-v1-4
```

```python
from diffusers import StableDiffusionPipeline
pipe = StableDiffusionPipeline.from_pretrained(top_model_id)
```

Available tasks

- Documented with a **webpage**
- Also available via JSON:
  `https://huggingface.co/api/tasks`

```python
import json
from urllib.request import urlopen
url = "https://huggingface.co/api/tasks"

with urlopen(url) as url:
    tasks = json.load(url)
print(tasks.keys())
```

```
dict_keys(['any-to-any',
'audio-classification',
'audio-to-audio', ...'])
```

[1] https://huggingface.co/tasks [2] https://huggingface.co/api/tasks

Let's practice!

MULTI-MODAL MODELS WITH HUGGING FACE

Preprocessing different modalities

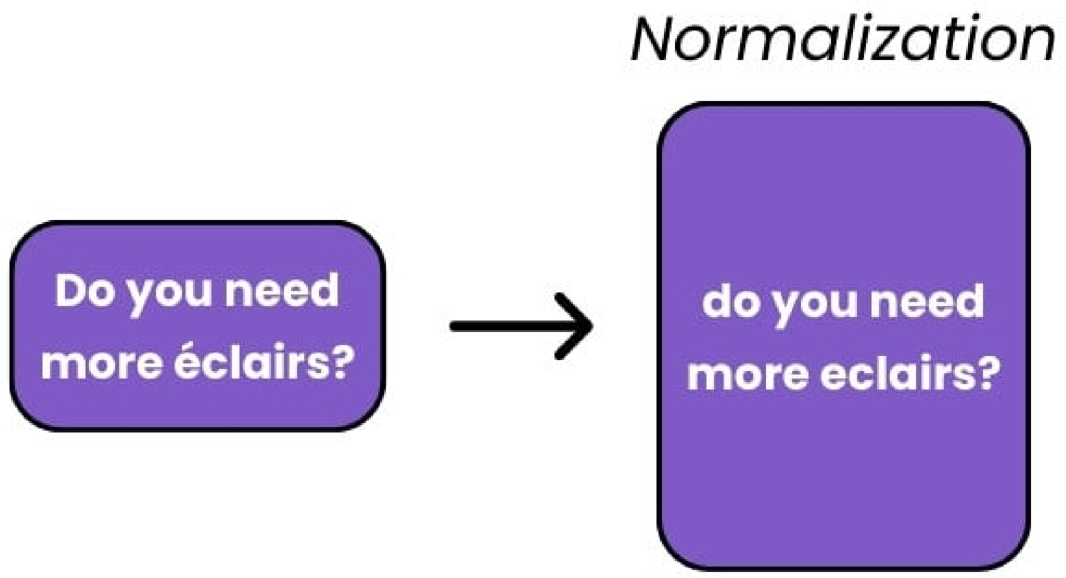
MULTI-MODAL MODELS WITH HUGGING FACE

James Chapman
Curriculum Manager, DataCamp

Preprocessing text

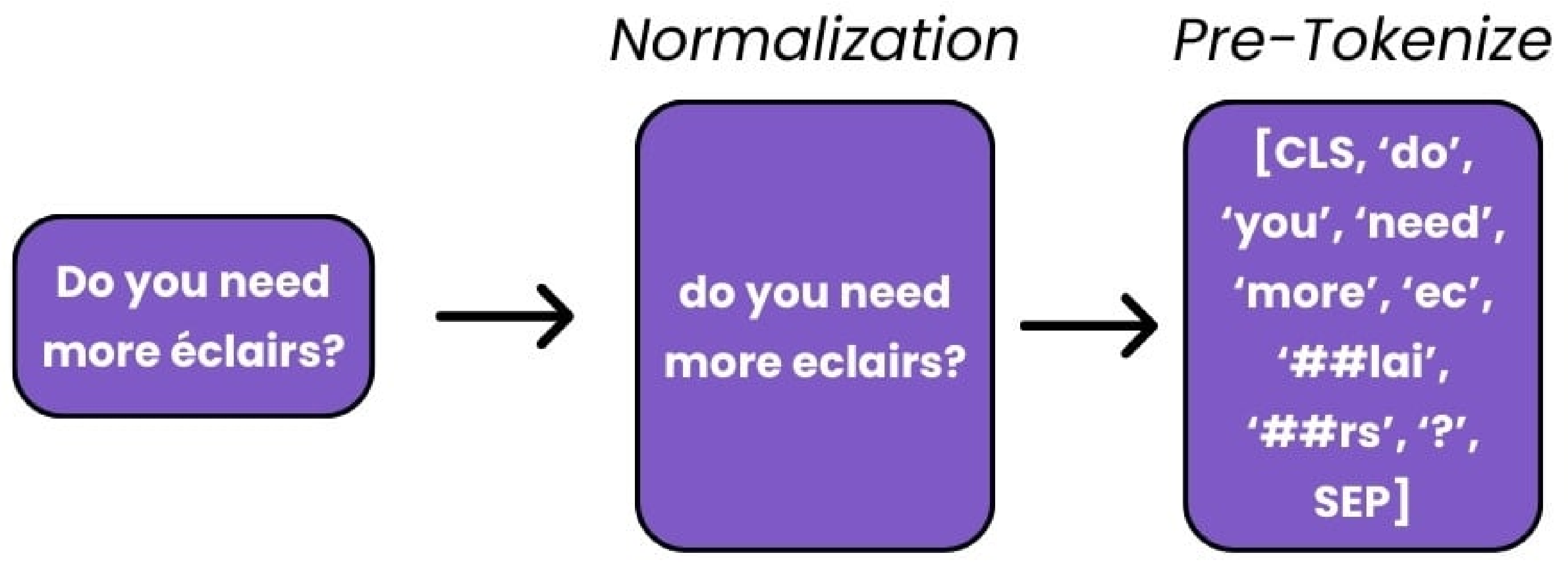
- **Tokenizer:** maps text → model input

Preprocessing text



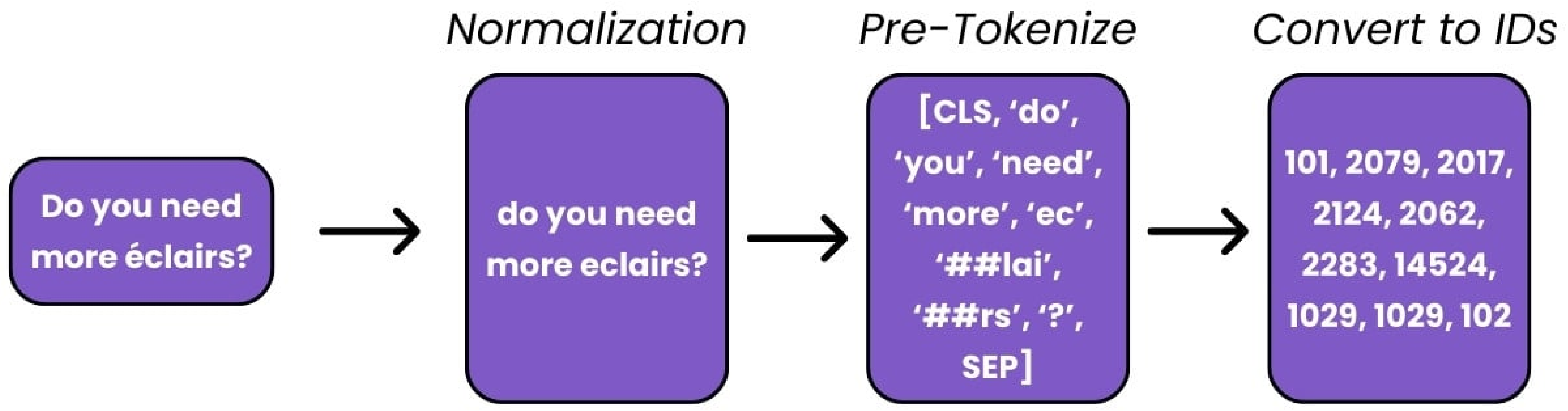
- **Tokenizer:** maps text → model input
  - **Normalization:** lowercasing, removing special characters, removing extra whitespace

Preprocessing text


Normalization
Pre-Tokenize
Do you need more éclairs?
do you need more eclairs?
[CLS, 'do', 'you', 'need', 'more', 'ec', '##lai', '##rs', '?', SEP]

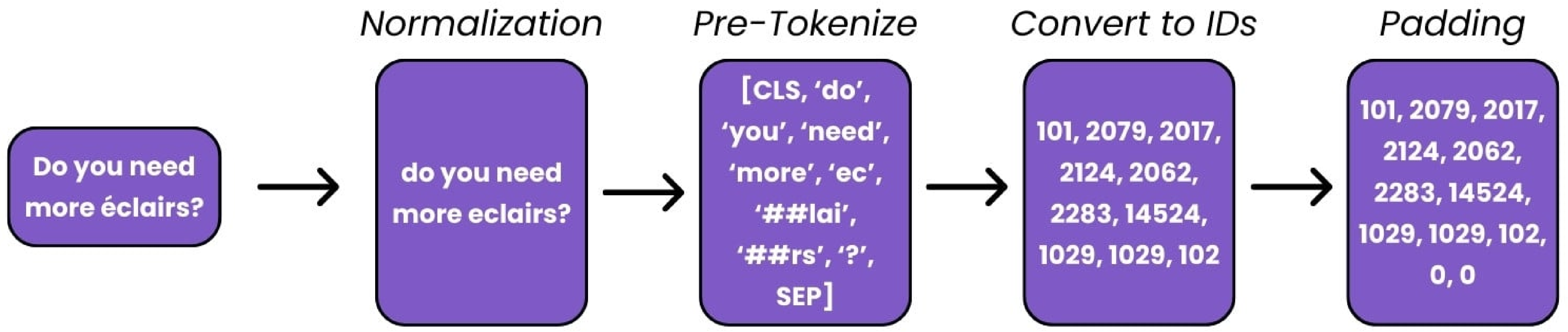
- **Tokenizer:** maps text → model input
  - **Normalization:** lowercasing, removing special characters, removing extra whitespace
  - **(Pre-)tokenization:** splitting text into words/subwords

Preprocessing text


Normalization
Pre-Tokenize
Convert to IDs
Do you need more éclairs?
do you need more eclairs?
[CLS, 'do', 'you', 'need', 'more', 'ec', '##lai', '##rs', '?', SEP]
101, 2079, 2017, 2124, 2062, 2283, 14524, 1029, 1029, 102

- **Tokenizer:** maps text → model input
  - **Normalization:** lowercasing, removing special characters, whitespace
  - **(Pre-)tokenization:** splitting text into words/subwords
  - **ID conversion:** Mapping of tokens to integers using a vocabulary

Preprocessing text



- **Tokenizer:** maps text → model input
  - **Normalization:** lowercasing, removing special characters, whitespace
  - **(Pre-)tokenization:** splitting text into words/subwords
  - **ID conversion:** Mapping of tokens to integers using a vocabulary
  - **Padding:** Adding additional tokens for consistent length

Preprocessing text

```python
from transformers import AutoTokenizer
tokenizer = AutoTokenizer.from_pretrained('distilbert/distilbert-base-uncased')
text = "Do you need more éclairs?"

print(tokenizer.backend_tokenizer.normalizer.normalize_str(text))
```

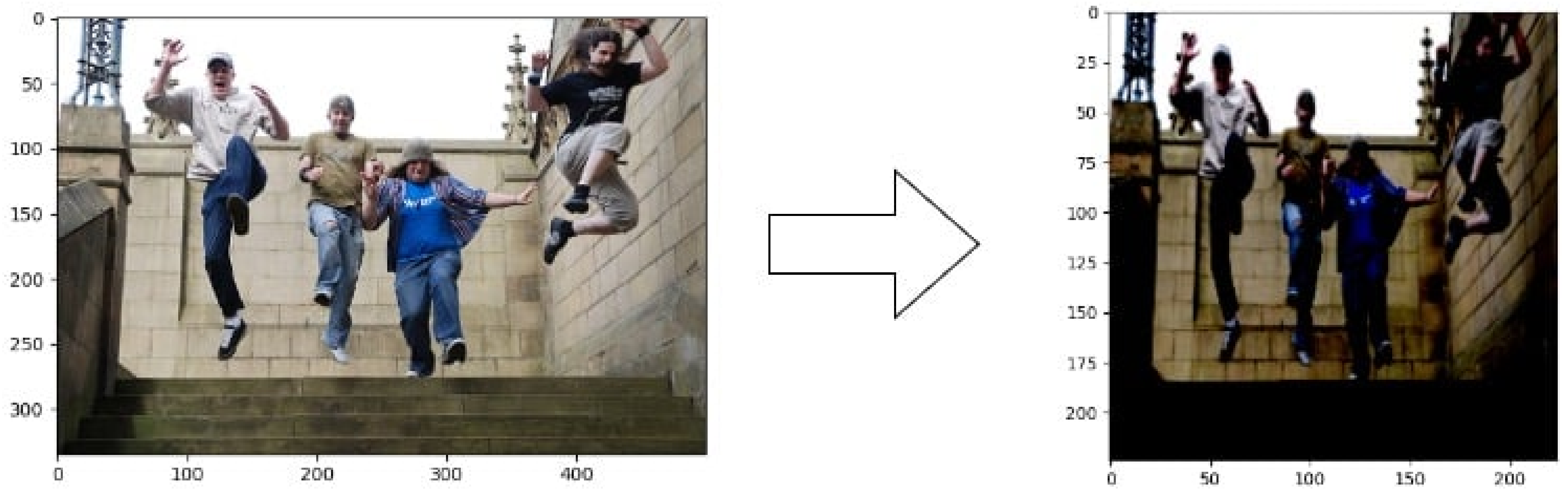
```
do you need more eclairs
```

```python
tokenizer(text, return_tensors='pt', padding=True)
```

```
{'input_ids': tensor([[  101,  ..., 102]]), ...}
```

Preprocessing images

- **Normalization:** pixel intensity updates
- **Resize:** Match input layer of model
- **General rule** → use preprocessing of original model



[1] https://huggingface.co/datasets/nlphuji/flickr30k

Preprocessing images

Multimodal tasks require *consistent preprocessing*:

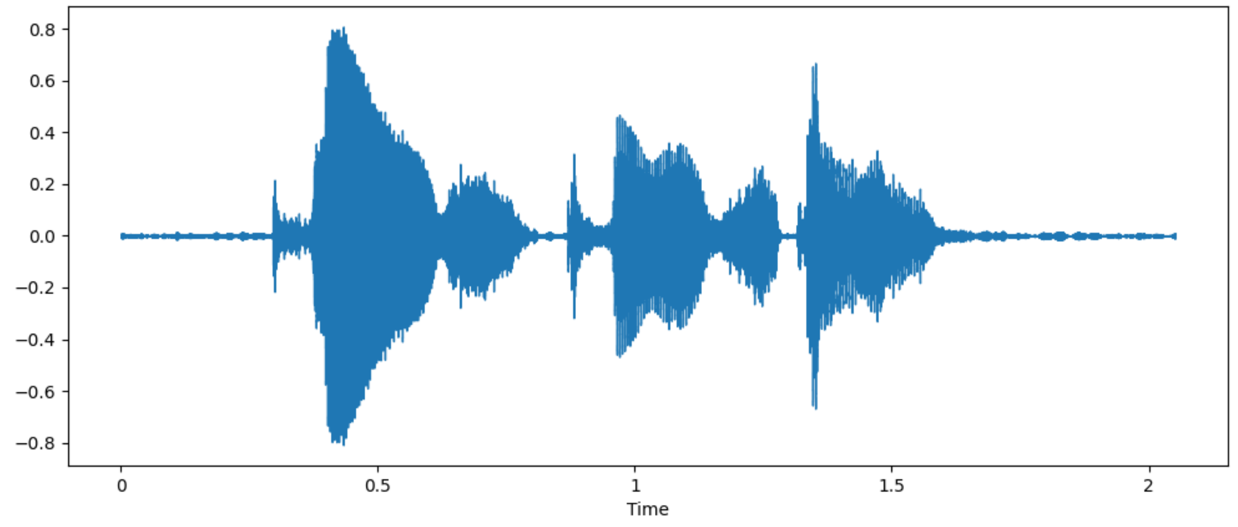
```python
from transformers import BlipProcessor, BlipForConditionalGeneration
checkpoint = "Salesforce/blip-image-captioning-base"
model = BlipForConditionalGeneration.from_pretrained(checkpoint)
processor = BlipProcessor.from_pretrained(checkpoint)
```

Encode image → transform to text encoding → decode text

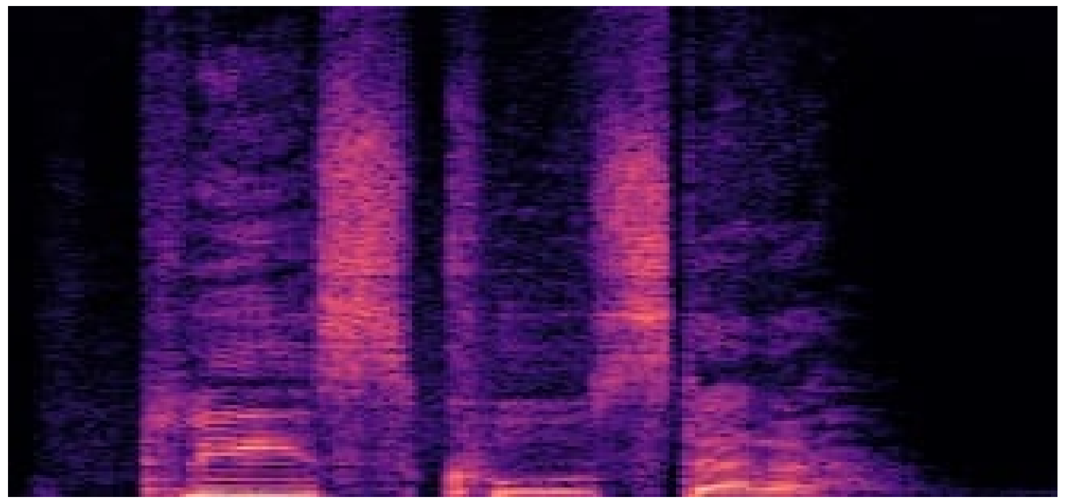
```python
image = load_dataset("nlphuji/flickr30k")['test'][11]["image"]
inputs = processor(images=image, return_tensors="pt")
output = model.generate(**inputs)
print(processor.decode(output[0]))
```

```
[{'generated_text': 'a group of people jumping'}]
```

Preprocessing audio



Feature extraction as model input (**spectrogram**)



- Audio preprocessing:
  - Sequential array → filter/padding
  - Sampling rate → resampling

Preprocessing audio

```python
from datasets import load_dataset, Audio
dataset = load_dataset("CSTR-Edinburgh/vctk")["train"]
dataset = dataset.cast_column("audio", Audio(sampling_rate=16_000))
```

- Model specific full preprocessors should be available:

```python
from transformers import AutoProcessor
processor = AutoProcessor.from_pretrained("openai/whisper-small")
audio_pp = processor(dataset[0]["audio"]["array"],
                     sampling_rate=16_000, return_tensors="pt")
```

- *Sampling rate must match model input requirements*

[1] https://huggingface.co/datasets/CSTR-Edinburgh/vctk

Let's practice!

MULTI-MODAL MODELS WITH HUGGING FACE

Pipeline tasks and evaluations

MULTI-MODAL MODELS WITH HUGGING FACE

James Chapman
Curriculum Manager, DataCamp

Pipelines vs. model components

Current approach

```python
from transformers import BlipProcessor, BlipForConditionalGeneration
checkpoint = "Salesforce/blip-image-captioning-base"
processor = BlipProcessor.from_pretrained(checkpoint)
model = BlipForConditionalGeneration.from_pretrained(checkpoint)
```

Pipelines

```python
from transformers import pipeline
pipe = pipeline("image-to-text", model=checkpoint)
```

Example comparison

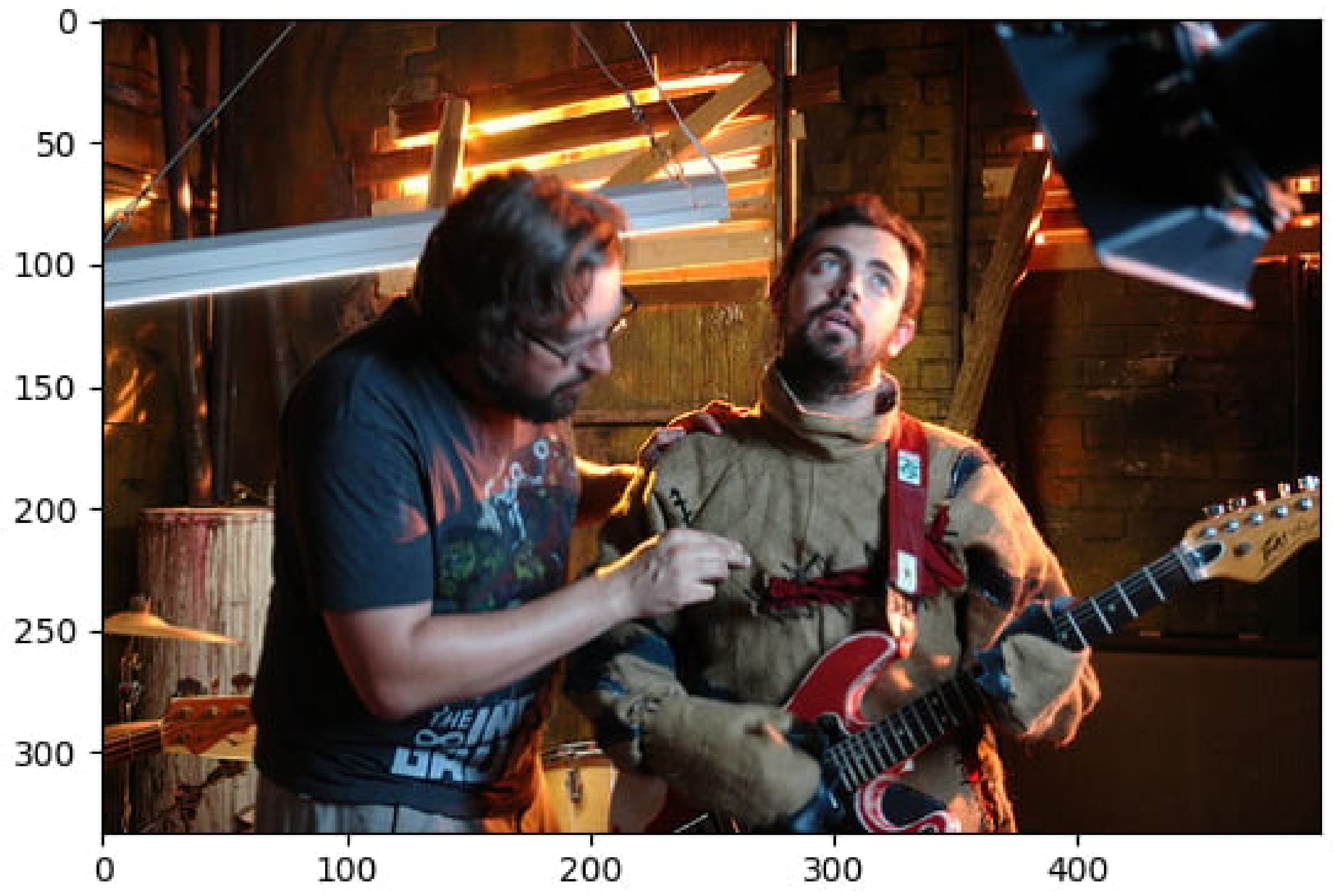
Preprocessor and model directly

```
inputs = processor(images=image,
                   return_tensors="pt")
gen = model.generate(**inputs)
processor.decode(gen[0])
```

Pipeline

```
pipe(image)
```

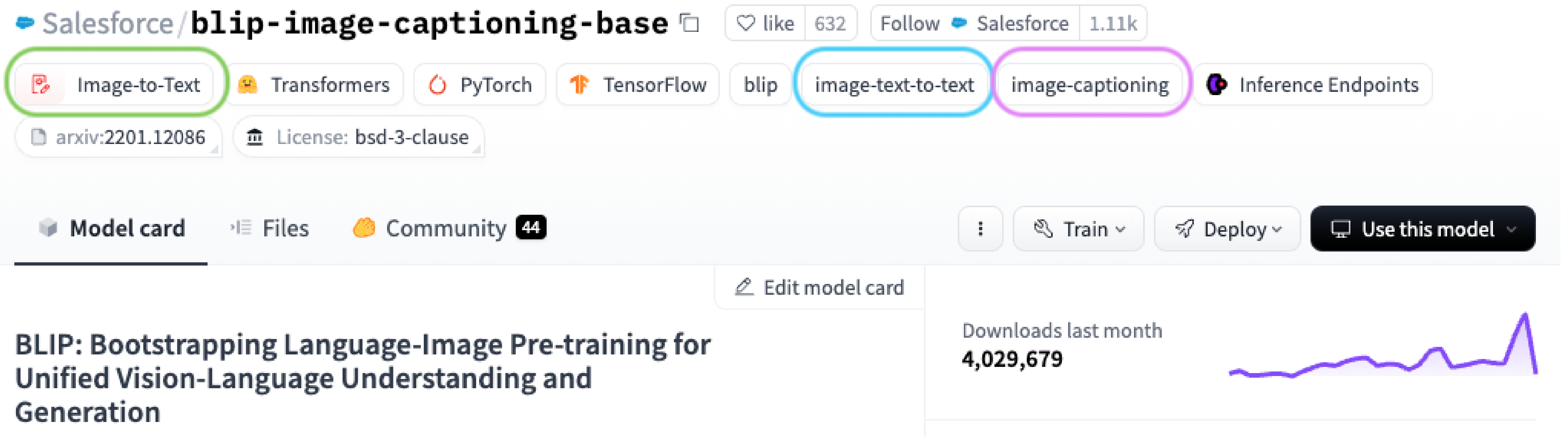
```
[{'generated_text':
'a man wearing a black shirt'}]
```



Finding models and tasks

Find models for a pipeline via the API:

```
from huggingface_hub import HfApi
model = list(api.list_models(task="text-to-image", limit=5))
pipe = pipeline("text-to-image", model[0].id)
```



Salesforce/**blip-image-captioning-base** like 632 Follow Salesforce 1.11k

Image-to-Text | Transformers | PyTorch | TensorFlow | blip | image-text-to-text | image-captioning | Inference Endpoints

arxiv:2201.12086 | License: bsd-3-clause

BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation

Downloads last month
4,029,679

Passing options to models

- `MusicgenForConditionalGeneration` under-the-hood

```
pipe = pipeline(task="text-to-audio",
                model="facebook/musicgen-small", framework="pt")
```

```
generate_kwargs = {"temperature": 0.8, "max_new_tokens": 20}
outputs = pipe("Classic rock riff", generate_kwargs=generate_kwargs)
```

- `temperature` (0-1): control randomness and creativity
- `max_new_tokens` : limit number of generated tokens

Evaluating pipeline performance

- **Accuracy:** total proportion of correct classifications
- **Precision:** how often class predictions are correct
- **Recall:** how many actual classes were correctly identified
- **F1 Score:** combines precision and recall

```
from evaluate import evaluator
task_evaluator = evaluator("image-classification")
metrics_dict = {
    "precision": "precision",
    "recall": "recall",
    "f1": "f1",
}
label_map = pipe.model.config.label2id
```

Evaluating pipeline performance

```
eval_results = task_evaluator.compute(
  model_or_pipeline=pipe,
  data=dataset,
  metric=evaluate.combine(metrics_dict),
  label_mapping=label_map)
```

```
print(eval_results)
```

```
{'precision': 0.999001923076923,
'recall': 0.999,
'f1': 0.9989999609405906, ...}
```

```
pipe = pipeline(task="image-classification",
model="ideepankarsharma2003/AI_ImageClassi
fication_MidjourneyV6_SDXL"
)
dataset = load_dataset("ideepankarsharma2003/
Midjourney_v6_Classification_small_shuffled")
```

Let's practice!

MULTI-MODAL MODELS WITH HUGGING FACE