“Deploy Hugging Face Transformers on AWS in just a few lines of code“

Talent Garden Cosenza

Nicola Procopio

30/05/2023

About me

Nicola Procopio

Senior Data Scientist

Contacts

nicola.procopio@finconsgroup.com

https://it.linkedin.com/in/nicolaprocopio

https://github.com/nickprock

https://huggingface.co/nickprock

Actually @

FINCONS GROUP

Background

healthware international

integris

ICAR CNR

Affare

KDD Lab

UNIVERSITÀ DELLA CALABRIA

Community

AI DEMOCRATIZE

tag Talent Garden Cosenza

iaml ITALIAN ASSOCIATION FOR MACHINE LEARNING

GDG

SC STARTUP CALABRIA

What's Hugging Face?

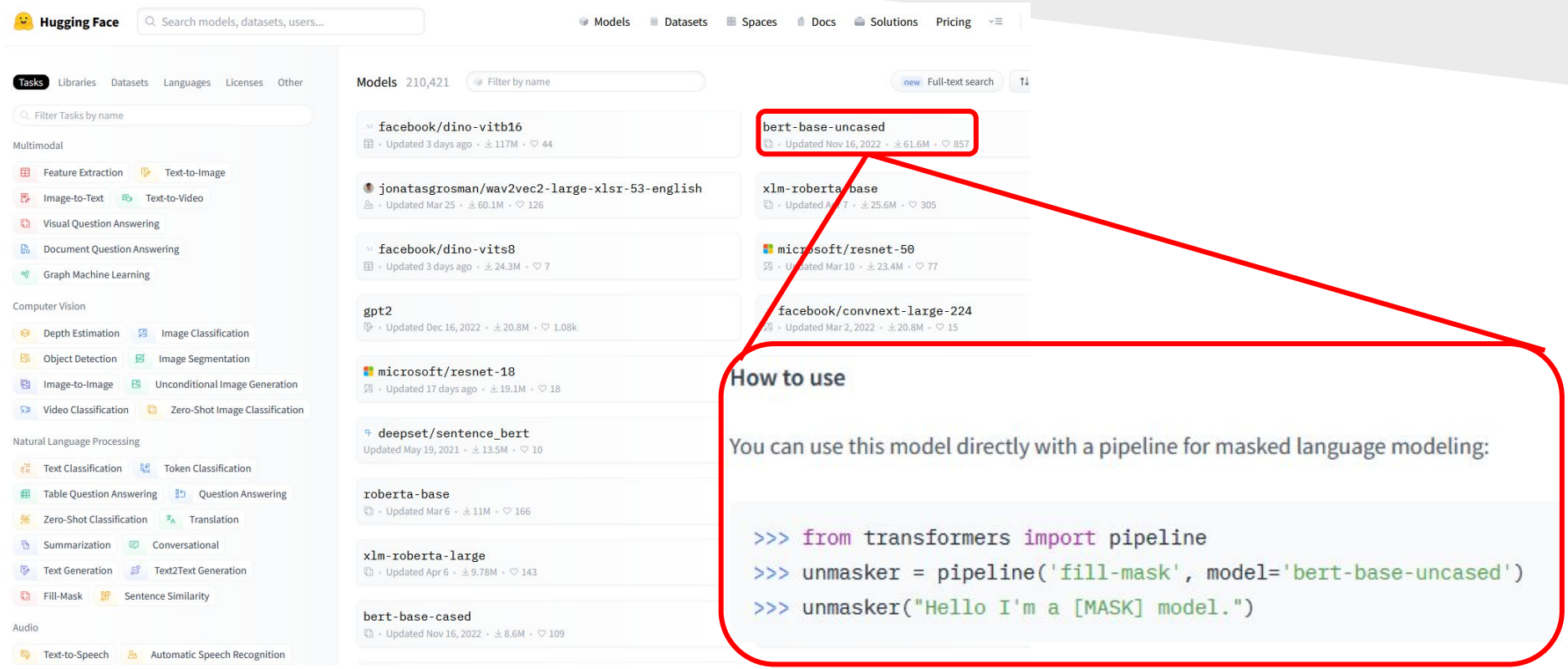
*"We are on a mission to democratize **good** machine learning, one commit at a time"*

- **Hugging Face** is the collaboration platform for the machine learning community.
- **The Hugging Face Hub** works as a central place where anyone can share, explore, discover, and experiment with open-source ML.
- With the fast-growing community, some of the most used **open-source ML libraries** and tools, and a talented science team exploring the edge of tech, **Hugging Face is at the heart of the AI revolution.**

★ Models: > 210K
★ Datasets: > 36K

★ Transformers: 102K stars on GitHub
★ Datasets: > 16.3K
★ Diffusers: > 14K
★ …

Hub & Libraries



Why Hugging Face on AWS?

Hugging Face Deep Learning Containers are available on AWS Sagemaker beacuse there are many advantages:

- **One command is all you need**
- **Accelerate machine learning from science to production (AWS Toolkit)**
- **Built-in performance**
- **Costs**

On Sagemaker you can:

- train
- deploy
- accelerate
- …

your models

```
from sagemaker.huggingface import HuggingFaceModel


# create Hugging Face Model Class and deploy it as SageMaker Endpoint
huggingface_model = HuggingFaceModel(...).deploy()
```

Focus on Deploy

Deploy an Hugging Face model is easy with Inference Toolkit.
First step install and setup the environment.

```
pip install sagemaker --upgrade
```

Sagemaker Environment

```
import sagemaker
sess = sagemaker.Session()
role = sagemaker.get_execution_role()
```

Local Environment

```
import sagemaker
import boto3

iam_client = boto3.client('iam')
role = iam_client.get_role(RoleName='role-name-of-your-iam-role-with-right-permissions')['Role']['Arn']
sess = sagemaker.Session()
```

Deploy from Model Hub

- Choose your model from Hugging Face Hub and create a model config.
  - {HF_MODEL_ID: checkpoint, HF_TASK: text-classification}

- Create your model class

```
# create Hugging Face Model Class
huggingface_model = HuggingFaceModel(
   env=hub,
   role=role,
   transformers_version="4.26",
   pytorch_version="1.13",
   py_version='py39',
)
```

- Deploy the endpoint

```
# deploy model to SageMaker Inference
predictor = huggingface_model.deploy(
   initial_instance_count=1,
   instance_type="ml.m5.xlarge"
)
```

REMEMBER!
Each HF_TASK has an inputs format!!!

Deploy from S3

- Create a *.tar.gz* file contains your model
- Create your role and model class

```
import sagemaker
import boto3

try:
    role = sagemaker.get_execution_role()
except ValueError:
    iam = boto3.client('iam')
    role = iam.get_role(RoleName='sagemaker_execution_role')['Role']['Arn']

print(f"sagemaker role arn: {role}")
```

```
from sagemaker.huggingface import HuggingFaceModel

# create Hugging Face Model Class
huggingface_model = HuggingFaceModel(
    model_data="s3://hf-sagemaker-inference/model.tar.gz",  # path to your trained sagemaker model
    role=role, # iam role with permissions to create an Endpoint
    transformers_version="4.26", # transformers version used
    pytorch_version="1.13", # pytorch version used
    py_version="py39", # python version of the DLC
)
```

Serverless Inference

- Deploy and scale ML models in a Serverless fashion
- Serverless endpoints automatically launch compute resources and scale them in and out depending on traffic similar to AWS Lambda.
- Serverless Inference is ideal for workloads which have idle periods between traffic spurts and can tolerate cold starts.
- Pay-per-use model
- Some Limitations:
  - Memory size
  - Concurrent Invocations
  - Cold Start

Serverless Inference

How the code changes?

Only import a module,
the object initialization and
a few parameters settings

```
from sagemaker.huggingface.model import HuggingFaceModel
from sagemaker.serverless import ServerlessInferenceConfig

# Hub Model configuration. <https://huggingface.co/models>
hub = {
    'HF_MODEL_ID':'distilbert-base-uncased-finetuned-sst-2-english',
    'HF_TASK':'text-classification'
}

# create Hugging Face Model Class
huggingface_model = HuggingFaceModel(
   env=hub,                      # configuration for loading model from Hub
   role=role,                    # iam role with permissions to create an Endpoint
   transformers_version="4.26",  # transformers version used
   pytorch_version="1.13",        # pytorch version used
   py_version='py39',            # python version used
)

# Specify MemorySizeInMB and MaxConcurrency in the serverless config object
serverless_config = ServerlessInferenceConfig(
    memory_size_in_mb=4096, max_concurrency=10,
)

# deploy the endpoint endpoint
predictor = huggingface_model.deploy(
    serverless_inference_config=serverless_config
)
```

Deploy Custom Models

- You need to use libraries not directly supported by the toolkit (es. **SetFit**, Sentence Transformers, SpanMarker, …)
- Your output format isn’t the default
- Other reasons why you cannot use the inference endpoint as it is

Deploy Custom Models

In our model folder we must add:

1. requirements.txt
2. handler.py
3. code/inference.py

Create a deploy.py, the script which is creating your endpoint.

requirements.txt is the file that contains the dependencies, in my case it had only one line: ***setfit.***
For S3 endpoint put it into code folder.

Deploy Custom Models

handler.py defines:

- how the endpoint must **load the model**
- what the **input** looks like
- what should return as **output**

```python
from typing import Dict, List, Any
from setfit import SetFitModel


class EndpointHandler:
    def __init__(self, path=""):
        # load model
        self.model = SetFitModel.from_pretrained(path)
        # ag_news id to label mapping
        self.id2label = {0: "World", 1: "Sports", 2: "Business", 3: "Sci/Tech"}

    def __call__(self, data: Dict[str, Any]) -> List[Dict[str, Any]]:
        """
         data args:
              inputs (:obj: `str`)
        Return:
              A :obj:`list` | `dict`: will be serialized and returned
        """
        # get inputs
        inputs = data.pop("inputs", data)
        if isinstance(inputs, str):
            inputs = [inputs]

        # run normal prediction
        scores = self.model.predict_proba(inputs)[0]

        return [{"label": self.id2label[i], "score": score.item()} for i, score in enumerate(scores)]
```

Deploy Custom Models

The **code folder** contains ***inference.py:***

- model_fn

```
def model_fn(model_dir):
    model = SetFitModel.from_pretrained(model_dir)
    return model
```

- input_fn

```
def input_fn(input_data, content_type):
    decoded_input_data = decoder_encoder.decode(input_data, content_type)
    return decoded_input_data
```

- predict_fn

```
def predict_fn(data, model):
    setFit_model = model

    inputs = data.pop("inputs", data)
    if isinstance(inputs, str):
        inputs = [inputs]

    # run normal prediction
    scores = model.predict_proba(inputs)[0]

    return [{"label": model_id2label[i], "score": round(score.item(),4)} for i, score in enumerate(scores)]
```

- output_fn → decode the result

Deploy Custom Models

Create the endpoint, file ***deploy.py***:

```
hf_model = HuggingFaceModel(
        model_data="s3://call-your-model/model1.tar.gz",
        role=role,
        transformers_version="4.6.1",
        pytorch_version="1.7.1",
        source_dir="code",
        py_version="py36",
        source_dir="model1/code"
        entry_point="inference.py",
    )
```

```
def predict(input):
    predictor = HuggingFacePredictor(
        endpoint_name="huggingface-endpoint")
    resp = predictor.predict({"inputs": input})
    print(resp)
    out = [data['label'] for data in resp if data['score'] >= 0.5]
    return out
```

```
predictor = huggingface_model.deploy(
    serverless_inference_config = ServerlessInferenceConfig(memory_size_in_mb=6144),
)
```

Notes

- **Code repository:**
  - *Hugging Face Sagemaker workshop series*
  - *Hugging Face notebooks on Sagemaker*

- **Follow Philipp Schmid (Hugging Face ML Eng & AWS ML Hero):**
  - *github*
  - *website*

- **Blog:**
  - *The Partnership: Amazon SageMaker and Hugging Face*
  - *Deploy Hugging Face models easily with Amazon SageMaker*

Thanks!

LET'S CHAT!

Remember to join  Hugging Face