A Guided Tour Through the Transformers Landscape

SwissText – 15 June 2021
NLP with Transformers

Hugging Face
Solving NLP one commit at a time!

https://huggingface.co

Repositories 200 Packages 44 People 89 Teams 3 Projects 4 Sponsoring 4 Settings

Pinned repositories

transformers
Transformers: State-of-the-art Natural Language Processing for PyTorch and TensorFlow 2.0.
Python 6.4k 10.7k

datasets
The largest hub of ready-to-use NLP datasets for ML models with fast, easy-to-use and efficient data manipulation tools.
Python 7.3k 861

~ awesome-papers ~
Research & presentation materials from Hugging Face's internal science day.
1.8k 154

accelerate
A simple way to train and use PyTorch models with multi-GPU, TPUs, mixed-precision.
Python 160 14

huggingface_hub
Client library to download and publish models and other files on the Hugging Face hub.
Python 45 7

Education

Open Source
A brief history

4 years old this month

Attention Is All You Need

A brief history

4 years old this month

A “Cambrian explosion”
Main ingredients

Transformer Blocks

Image credit: Lin et al (2021)
Main ingredients

Transformer Blocks

Language Modelling

Image credit: Lin et al (2021)

Masked LM

Autoregressive LM
The modern paradigm
The modern paradigm
Works for vision too 😶

Image GPT (iGPT)

Image credit: Chen et al (2020)

Vision Transformer (ViT)

Image credit: Dosovitskiy et al (2020)
Including multiple modalities

**Text Prompt:** an illustration of a baby daikon radish in a tutu walking a dog

**AI-Generated Images:**

- ![Image 1](image1.png)
- ![Image 2](image2.png)
- ![Image 3](image3.png)
- ![Image 4](image4.png)

**Text Prompt:** an armchair in the shape of an avocado...

**AI-Generated Images:**

- ![Image 5](image5.png)
- ![Image 6](image6.png)
- ![Image 7](image7.png)
- ![Image 8](image8.png)

**A multimodal document as query:**

**Results more rely on**

- **Text:** 0.5
- **Image:** 0.5

**Search!**

Multimodal document search by Jina
Transformers are now everywhere

- NLP
- Speech
- Vision
- BioChem
- Time Series
- RL
Transformers are now everywhere

NLP
Speech
Vision
BioChem
Time Series
RL
Domain X
Bridging the science – industry divide

Humble Data Scientist
Get the code and model weights?
The wild west of open-source

Explosion of pre-trained models: which one do I choose?
The wild west of open-source

Explosion of pretrained models: which one do I choose?

Different APIs, missing docs, reproducibility issues, ...
Can we do better?
The AI community building the future.

Build, train and deploy state of the art models powered by the reference open source in natural language processing.

More than 5,000 organizations are using Hugging Face

- Amazon
  - Company
  - 1 model
- Allen Institute for AI
  - Non-Profit
  - 51 models
- Microsoft
  - Company
  - 47 models
- Google AI
  - Company
  - 130 models
- Facebook AI
  - Company
  - 76 models
- Grammarly
  - Company
- Typeform
  - Company
  - 8 models
- asteroid-team
  - Non-Profit
Open-source @ Hugging Face

Hugging Face
Solving NLP, one commit at a time!

Repositories 2018 Packages People Teams Projects Sponsorship Settings

Pinned repositories

- **transformers**
  - Python 44.9k 10.7k

- **datasets**
  - The largest hub of ready-to-use NLP datasets for ML models with fast, easy-to-use and efficient data manipulation tools.
  - Python 7.3k 860

- **tokenizers**
  - Fast State-of-the-Art Tokenizers optimized for Research and Production.
  - Rust 4.5k 330

- **awesome-papers**
  - Papers & presentation materials from Hugging Face’s internal science day.
  - 1.8k 104

- **accelerate**
  - A simple way to train and use PyTorch models with multi-GPU, TPU, mixed-precision.
  - Python 306 14

- **huggingface_hub**
  - Client library to download and publish models and other files on the huggingface hub.
  - Python 85 7
The Hugging Face ecosystem

Model Hub → Load a model from the hub or initialize it from scratch → Tokenizers & Transformers

Tokenizers & Transformers → Train it using one or several datasets → Datasets

Datasets → Model Hub

Model Hub → Upload it on the hub once done
The Hugging Face ecosystem

Model Hub

Tokenizers & Transformers

Upload it on the hub once done

Datasets

Train it using one or several datasets

Load a model from the hub or initialize it from scratch
The Hugging Face ecosystem

Model Hub → Tokenizers & Transformers → Datasets → Model Hub

Load a model from the hub or initialize it from scratch.

Train it using one or several datasets.

Upload it on the hub once done.
Tokenizers and Transformers

```python
from transformers import AutoTokenizer, AutoModelForMaskedLM
tokenizer = AutoTokenizer.from_pretrained("bert-base-uncased")
model = AutoModelForMaskedLM.from_pretrained("bert-base-uncased")
```
Tokenizers & Transformers

- **50+** Model architectures
- **Used by more than 5,000** companies
- **Simple API across all architectures**
The Hugging Face ecosystem

Model Hub

Load a model from the hub or initialize it from scratch

Tokenizers & Transformers

Train it using one or several datasets

Datasets

Upload it on the hub once done

Model Hub
Datasets

The Hub:
- Largest hub of ready-to-use datasets
- 1000+ Datasets available
- 450+ languages and dialects supported
The python library:
- Load any dataset in one line
- Supports huge datasets without RAM limitations
- Fast iterations and querying
The Hugging Face ecosystem

Model Hub → Tokenizers & Transformers → Datasets → Model Hub

- Load a model from the hub or initialize it from scratch
- Train it using one or several datasets
- Upload it on the hub once done
from huggingface_hub import HfApi
api = HfApi()
Hugging Face Hub

class DummyModel(nn.Module, ModelHubMixin):
    def __init__(self, **kwargs):
        super().__init__()
        self.config = kwargs.pop("config", None)
        self.l1 = nn.Linear(2, 2)

    def forward(self, x):
        return self.l1(x)

model = DummyModel()
model.save_pretrained("my-dummy-model")
model.push_to_hub("my-dummy-model", organization="huggingface")

# Reload it from any device!
model = DummyModel.from_pretrained("huggingface/my-dummy-model")
- Python API
- Simple mixin for your PyTorch Module
- Supports Transformers, AllenNLP, Asteroid, Spacy, Timm ...
The Hugging Face ecosystem

- Transformers
- Datasets
- Model Hub
- Inference API

- Initialize it from scratch
- Upload it on the hub once done
- Train it using one or several datasets
- Other deployment options
Inference API

Fill-Mask

The goal of life is [MASK].

This model can be loaded on the Inference API on demand.
Computation time on cpu cached.

<table>
<thead>
<tr>
<th>word</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>happiness</td>
<td>0.836</td>
</tr>
<tr>
<td>survival</td>
<td>0.831</td>
</tr>
<tr>
<td>salvation</td>
<td>0.017</td>
</tr>
<tr>
<td>freedom</td>
<td>0.017</td>
</tr>
<tr>
<td>unity</td>
<td>0.015</td>
</tr>
</tbody>
</table>

Token Classification

My name is Clara and I live in Berkeley, California. I work at this cool company called Hugging Face. Inc.

This model can be loaded on the Inference API on demand.
Computation time on cpu cached.

<table>
<thead>
<tr>
<th>word</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clara</td>
<td>PER</td>
</tr>
<tr>
<td>Berkeley</td>
<td>LOC</td>
</tr>
<tr>
<td>California</td>
<td>LOC</td>
</tr>
<tr>
<td>Hugging Face</td>
<td>ORG</td>
</tr>
</tbody>
</table>

)}) JSON Output
The Hugging Face ecosystem

Model Hub → Transformers → Model Hub

Load a model from the hub or initialize it from scratch

Transformers

Train it using one or several datasets

Upload it on the hub once done

Datasets
Three ways to train a model
from transformers import AutoModelForSequenceClassification, Trainer, TrainingArguments
model = AutoModelForSequenceClassification.from_pretrained("bert-base-uncased")
training_args = TrainingArguments(
    output_dir="./my-finetuned-model", # output directory
    num_train_epochs=3, # total number of training epochs
    per_device_train_batch_size=16, # batch size per device during training
    per_device_eval_batch_size=64, # batch size for evaluation
    evaluation_strategy="epoch", # evaluate every epoch
    weight_decay = 0.01, # strength of weight decay
)
trainer = Trainer(
    model=model, # model to be trained
    args=training_args, # training arguments, defined above
    train_dataset=train_dataset, # the training set
    eval_dataset=test_dataset, # the evaluation set
    tokenizer=tokenizer, # the tokenizer used for preporcessing
)
train.train()
train.push_to_hub()
```python
import torch
import torch.nn.functional as F
from datasets import load_dataset
+ from accelerate import Accelerator

device = 'cpu'
+ device = accelerator.device

model = torch.nn.Transformer().to(device)
optim = torch.optim.Adam(model.parameters())

data = torch.utils.data.DataLoader(dataset, shuffle=True)
+ model, optim, data = accelerator.prepare(model, optim, data)

model.train()
for epoch in range(10):
    for source, targets in data:
        source = source.to(device)
        targets = targets.to(device)

        optimizer.zero_grad()

        output = model(source)
        loss = F.cross_entropy(output, targets)

- loss.backward()
+ accelerator.backward(loss)

        optimizer.step()
```
Accelerate: handling devices

```python
import torch
import torch.nn.functional as F
from datasets import load_dataset

+ from accelerate import Accelerator

+ accelerator = Accelerator()
- device = 'cpu'

- model = torch.nn.Transformer().to(device)
+ model = torch.nn.Transformer()

optim = torch.optim.Adam(model.parameters())

dataset = load_dataset('my_dataset')
data = torch.utils.data.DataLoader(dataset, shuffle=True)

+ model, optim, data = accelerator.prepare(model, optim, data)

model.train()
for epoch in range(10):
    for source, targets in data:
        - source = source.to(device)
        - targets = targets.to(device)

        optimizer.zero_grad()

        output = model(source)
        loss = F.cross_entropy(output, targets)

        - loss.backward()
        + accelerator.backward(loss)

        optimizer.step()
```
Currently supported:

- Training
- Evaluation
- CPU, GPU, Multi GPU, TPU, Mixed precision
- AWS Sagemaker

Upcoming:

- Horovod
- FairScale
- DeepSpeed
- Others? 😞
# Upload your model data
autonlp upload --project sentiment_detection --split train
  --col_mapping review:review,sentiment:target
  --files ~/datasets/train.csv

# Train your model
autonlp train --project sentiment_detection

# Use your model
curl -X POST
  -H "Authorization: Bearer API key:jeZrkpoqfjzioaRaerjlbRQeKykrop"
  -H "Content-Type: application/json"
  -d '{"inputs":"The goal of life is [MASK]"}'
  https://api-inference.huggingface.co/models/sentiment_detection
Big Science

bigscience.huggingface.co/
What is Big Science?

One-year research workshop on large multilingual datasets and large language models

Analogy with the Large Hadron Collider at CERN:

- has involved 10,000+ researchers
- from 100+ countries
- lead to the discovery of 59 hadrons
- publication of more than 2,800 papers 😱

In many scientific fields (epidemiology, space, fusion...), large-scale and worldwide research collaborations create tools useful for the entire research community, like the LHC, ITER, ISS...

Isn’t it time to build similar large, diverse, open research collaborations in AI/NLP as well?
Why do this?

**Research**
- Models not designed as general research tools
- Difficult involvement of academic researchers
- Lack of fields diversity of the research teams building them

**Environmental**
- Training parallel models in private setting => duplication of energy requirements
- Carbon footprint not documented/taken into account

**Ethical and societal**
- Shortcomings in the text corpora used to train these models
- Ethical/bias/usage question are usually asked a-posteriori
But large-scale public compute exists

Accelerated partition (or GPU partition)

- 261 four-GPU accelerated compute nodes with:
  - 2 Intel Cascade Lake 6248 processors (20 cores at 2.5 GHz), namely 40 cores per node
  - 192 GB of memory per node
  - 4 Nvidia Tesla V100 SXM2 GPUs (32 GB)
- 31 eight-GPU accelerated compute nodes, currently dedicated to the AI community with:
  - 2 Intel Cascade Lake 6226 processors (12 cores at 2.7 GHz), namely 24 cores per node
  - 20 nodes with 384 GB of memory and 11 nodes with 768 GB of memory
  - 8 Nvidia Tesla V100 SXM2 GPUs (32 GB)
- Extension in the summer of 2020, 351 four-GPU accelerated compute nodes with:
  - 2 Intel Cascade Lake 6248 processors (20 cores at 2.5 GHz), namely 40 cores per node
  - 192 GB of memory per node
  - 4 Nvidia Tesla V100 SXM2 GPUs (16 GB)

- Cumulated peak performance of 28 Pflop/s with a total of 2696 Nvidia V100 GPUs

Jean Zay supercomputer in France 😅
A brief history

- 🐥 **Early 2021**: Discussions between Thomas Wolf (HuggingFace), Stéphane Requena (GENCI) and Pierre-François Lavallée (IDRIS)

- 👪 **Very quickly**: Science team of HF + many members of the French academic and industrial AI and NLP research communities joined the discussion to further develop the project leading to the grant application

- 📝 **February 2021**: Grant application for 5 million GPU hours

- 🌍 **Following the grant submission**: open/extend to international research community

- 🛠 **When the project reached 200+ participants**: the organization of the project started to take shape and to adopt the structure of a research workshop

- 🚀 **19/04**: Grant accepted -- first half of the project

- 🌸 **28/04**: Kickoff event + project becomes public
Core research questions

- **Large models + large datasets**: exhibit intriguing and quite surprising behaviors from a research point of view

- Raise **many research questions** across many fields/subfields of AI/NLP:
  - **Fundamental**:
    - limits of what can be done with purely statistical and text-based approaches?
    - notion of what is an NLP task and what is the relation between a task and a dataset
  - **Bias/fairness**:
    - notion of bias and its relation to the dataset and training objectives
    - representativeness and stereotypes
    - memorization versus generalization and personal information memorization
  - **Environmental impact and carbon footprint**
  - And so many others (interpretability, relation to cognitive processes, use in linguistics...)
How can I participate?

- **General Advisor** (Steering Committee member):
  - *role* - give general scientific/organization advice - everyone here is in the SC by default
  - *time commitment* - *light* - reading a newsletter every 2 weeks - giving feedback/advice

- **Join a Working Group** (Organizing Committee member)
  - *role* - join one of the Working Groups to advise or participate (code, research...)
  - *time commitment* - *medium* - depends on the chosen WG

- **Chair/co-Chair a Working Group** (Organizing Committee member)
  - *role* - the chairs are responsible for providing the minimal amount of work necessary for having a barebone version of the task. If WG members are active, the chairs can mostly coordinate the effort and organise the decision process.
  - *time commitment* - *more significant* - depends on the chosen WG

- **A Workshop attendant** joining live events or some community events (tbd)
  - *role* - participating in the collaborative task following guidelines by the WG
  - *time commitment* - *free* - up to the attendant - open to anyone, beginners, people outside of the research fields, etc... - very accessible
Questions!