Escuela AI HUB de verano
Sobre IA y Educación

HANDS ON INTRODUCTION TO TRANSFORMERS

Jesús Cerquides (IIIA-CSIC)
(2/07/2024)

CSIC
Consejo Superior de Investigaciones Científicas
CSIC HUB Inteligencia Artificial
Fundación "la Caixa"
IFIC Institut de Física Corpuscular
Vniversitat de València

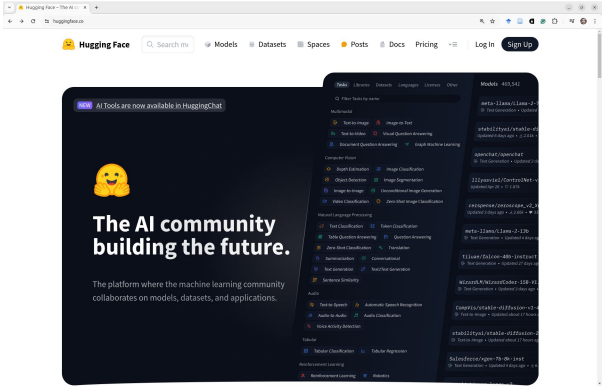
What we hope you will learn

- What is the Hugging face ecosystem and why it is interesting to you
- Run your first pipelines
- Go as far as you can on the Hugging Face NLP Course.

What is Hugging Face?

- GitHub of the ML world — a collaborative platform brimming with tools that empower anyone to create, train, and deploy NLP and ML models using open-source code.
- These models come already pre-trained!
- “The AI community building the future.”

Hugging face Hub



The Hugging Face Hub is a platform with over 350k models, 75k datasets, and 150k demo apps (Spaces), all open source and publicly available, in an online platform where people can easily collaborate and build ML together. The Hub works as a central place where anyone can explore, experiment, collaborate, and build technology with Machine Learning

Hugging face repositories

Repositories

Models, Spaces, and Datasets are hosted on the Hugging Face Hub as Git repositories, which means that version control and collaboration are core elements of the Hub. In a nutshell, a repository (also known as a repo) is a place where code and assets can be stored to back up your work, share it with the community, and work in a team.

You will start worrying about repositories if you want to submit your models one day. Today, you are just a user!

Hugging face models

The Model Hub is where the members of the Hugging Face community can host all of their model checkpoints for simple storage, discovery, and sharing. Download pre-trained models with the `huggingface_hub` client library, with 🤗 `Transformers` for fine-tuning and other usages or with any of the over 15 integrated libraries. You can even leverage the Serverless Inference API or Inference Endpoints. to use models in production settings.

Hugging face model cards

Model cards are files that accompany the models and provide handy information. Under the hood, model cards are simple Markdown files with additional metadata. Model cards are essential for discoverability, reproducibility, and sharing! You can find a model card as the `README.md` file in any model repo.

The model card should describe:

- the model
- its intended uses & potential limitations, including biases and ethical considerations as detailed in Mitchell, 2018
- the training params and experimental info (you can embed or link to an experiment tracking platform for reference)
- which datasets were used to train your model
- the model's evaluation results

The model card template is available here.

Hugging face datasets

The Hugging Face Hub hosts a large number of community-curated datasets for a diverse range of tasks such as translation, automatic speech recognition, and image classification. Alongside the information contained in the dataset card, many datasets, such as GLUE, include a Dataset Viewer to showcase the data.

Each dataset is a Git repository that contains the data required to generate splits for training, evaluation, and testing. For information on how a dataset repository is structured, refer to the Data files Configuration page. Following the supported repo structure will ensure that the dataset page on the Hub will have a Viewer.

There is also the 🤗 Datasets Python package!!!

Hugging face dataset cards

Each dataset may be documented by the `README.md` file in the repository. This file is called a dataset card, and the Hugging Face Hub will render its contents on the dataset's main page. To inform users about how to responsibly use the data, it's a good idea to include information about any potential biases within the dataset. Generally, dataset cards help users understand the contents of the dataset and give context for how the dataset should be used.

You can also add dataset metadata to your card. The metadata describes important information about a dataset such as its license, language, and size. It also contains tags to help users discover a dataset on the Hub, and data files configuration options. Tags are defined in a YAML metadata section at the top of the `README.md` file.

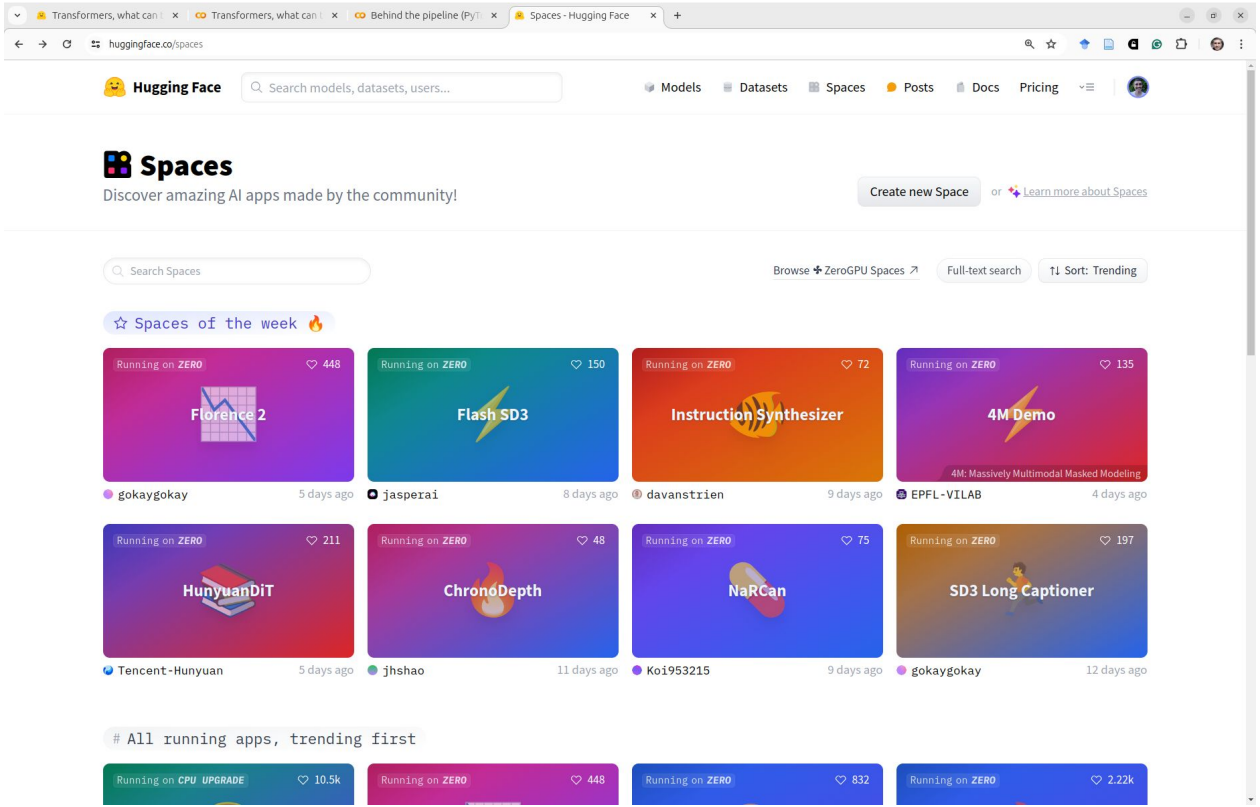
Hugging face spaces

Hugging Face Spaces offer a simple way to host ML demo apps directly on your profile or your organization's profile. This allows you to create your ML portfolio, showcase your projects at conferences or to stakeholders, and work collaboratively with other people in the ML ecosystem.

SDKs that let you build cool apps in Python in a matter of minutes: Streamlit and Gradio, but you can also unlock the whole power of Docker and host an arbitrary Dockerfile. Finally, you can create static Spaces using JavaScript and HTML.

You'll also be able to upgrade your Space to run on a GPU or other accelerated hardware. ⚡

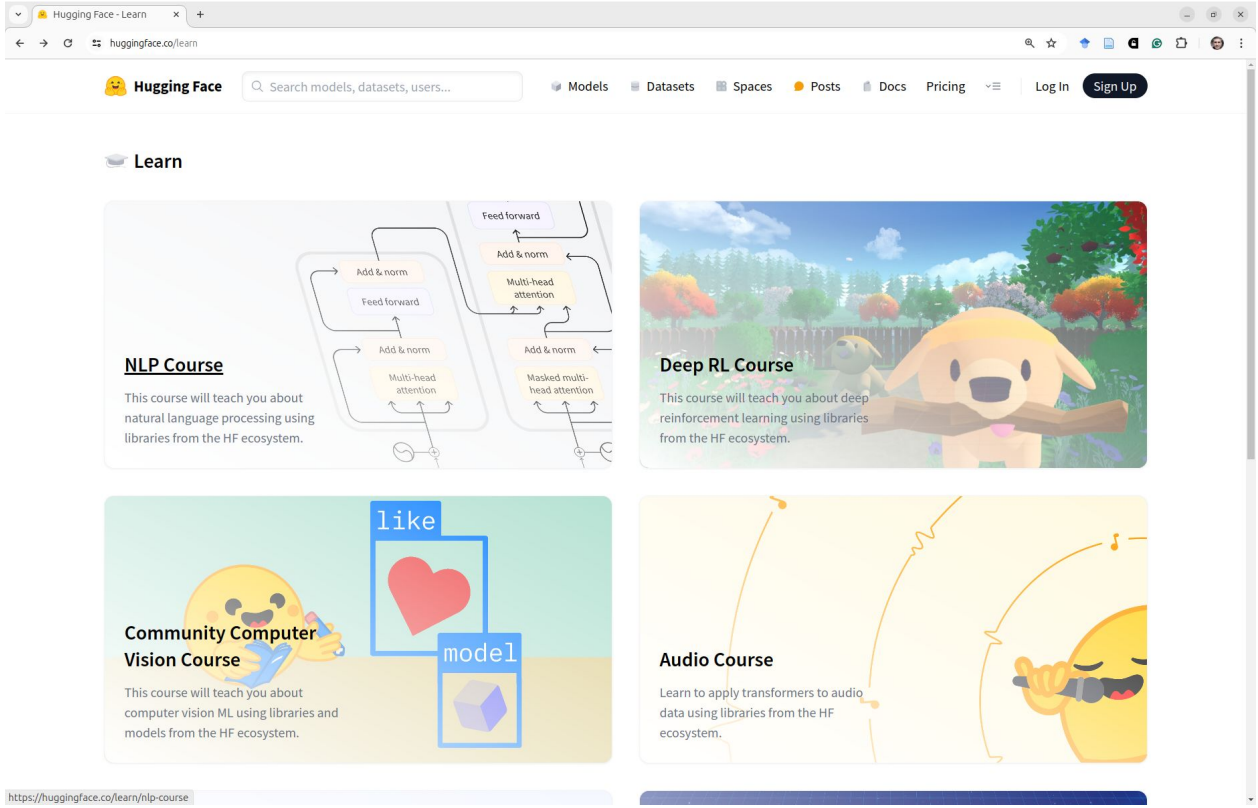
Hugging Face spaces (let’s play!)



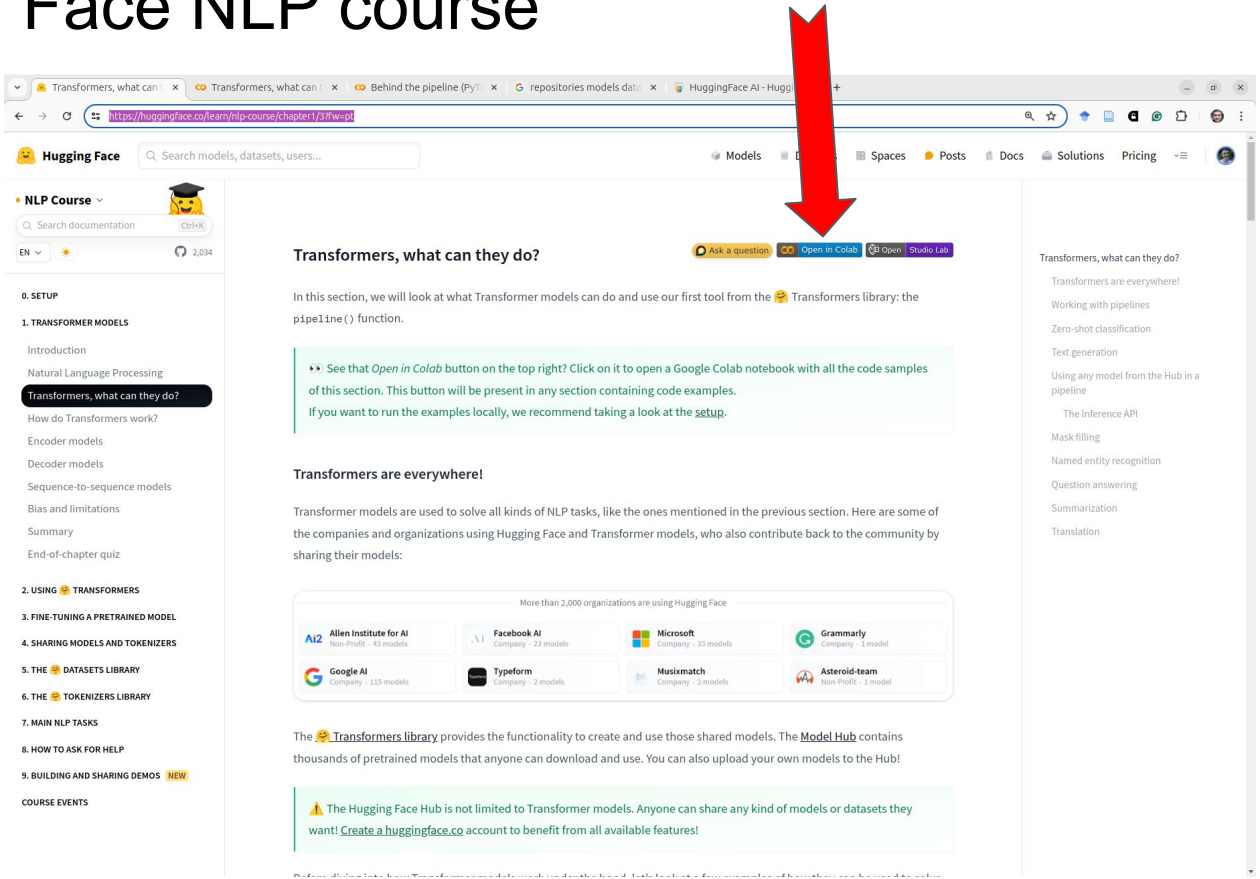
Recap

- What is the Hugging face ecosystem?
  - Tasks
  - Repositories
  - Models
  - Datasets
  - Spaces
- Run your first pipeline
-

Hugging Face resources



Hugging Face NLP course



Playing with Transformers (I)

1. Create your own zero-shot classifier to distinguish between sentences from of “Cervantes” vs “Lope de Vega”.
2. Cervantes:
   a. «Amistades que son ciertas nadie las puede turbar.»
   b. «Al bien hacer jamás le falta premio.»
   c. «Más vale el buen nombre que las muchas riquezas.»
3. Lope de Vega:
   a. «Que pobreza no es vileza mientras no hace cosas malas.»
   b. «Celos son hijos del amor, mas son bastardos, te confieso.»
   c. «¡Dios me libre de enemistades de amigos!»
4. Change the former zero-shot classifier to use the model `Recognai/bert-base-spanish-wwm-cased-xnli`

Playing with Transformers (II)

Text generation

Now let's see how to use a pipeline to generate some text. The main idea here is that you provide a prompt and the model will auto-complete it by generating the remaining text. This is similar to the predictive text feature that is found on many phones. Text generation involves randomness, so it's normal if you don't get the same results as shown below.

```
from transformers import pipeline

generator = pipeline("text-generation")
generator("In this course, we will teach you how to")
```

```
[{'generated_text': 'In this course, we will teach you how to understand and use '
                    'data flow and data interchange when handling user data. We '
                    'will be working with one or more of the most commonly used '
                    'data flows — data flows of various types, as seen by the '
                    'HTTP'}]
```

You can control how many different sequences are generated with the argument `num_return_sequences` and the total length of the output text with the argument `max_length`.

> ✏️ **Try it out!** Use the `num_return_sequences` and `max_length` arguments to generate two sentences of 15 words each.

Playing with Transformers (III)

Mask filling

The next pipeline you'll try is `fill-mask`. The idea of this task is to fill in the blanks in a given text:

```
from transformers import pipeline

unmasker = pipeline("fill-mask")
unmasker("This course will teach you all about <mask> models.", top_k=2)
```

```
[{'sequence': 'This course will teach you all about mathematical models.',
  'score': 0.19619831442832947,
  'token': 30412,
  'token_str': ' mathematical'},
 {'sequence': 'This course will teach you all about computational models.',
  'score': 0.04052725434303284,
  'token': 38163,
  'token_str': ' computational'}]
```

The `top_k` argument controls how many possibilities you want to be displayed. Note that here the model fills in the special `<mask>` word, which is often referred to as a *mask token*. Other mask-filling models might have different mask tokens, so it's always good to verify the proper mask word when exploring other models. One way to check it is by looking at the mask word used in the widget.

> ✏️ **Try it out!** Search for the `bert-base-cased` model on the Hub and identify its mask word in the Inference API widget. What does this model predict for the sentence in our `pipeline` example above?

Playing with Transformers (IV)

Named entity recognition

Named entity recognition (NER) is a task where the model has to find which parts of the input text correspond to entities such as persons, locations, or organizations. Let's look at an example:

```
from transformers import pipeline

ner = pipeline("ner", grouped_entities=True)
ner("My name is Sylvain and I work at Hugging Face in Brooklyn.")
```

```
[{'entity_group': 'PER', 'score': 0.99816, 'word': 'Sylvain', 'start': 11, 'end': 18},
 {'entity_group': 'ORG', 'score': 0.97960, 'word': 'Hugging Face', 'start': 33, 'end': 45},
 {'entity_group': 'LOC', 'score': 0.99321, 'word': 'Brooklyn', 'start': 49, 'end': 57}
]
```

Here the model correctly identified that Sylvain is a person (PER), Hugging Face an organization (ORG), and Brooklyn a location (LOC).

We pass the option `grouped_entities=True` in the pipeline creation function to tell the pipeline to regroup together the parts of the sentence that correspond to the same entity: here the model correctly grouped "Hugging" and "Face" as a single organization, even though the name consists of multiple words. In fact, as we will see in the next chapter, the preprocessing even splits some words into smaller parts. For instance, `Sylvain` is split into four pieces: `S`, `##yl`, `##va`, and `##in`. In the post-processing step, the pipeline successfully regrouped those pieces.

> ✏️ **Try it out!** Search the Model Hub for a model able to do part-of-speech tagging (usually abbreviated as POS) in English. What does this model predict for the sentence in the example above?

Playing with Transformers (V)

Translation

For translation, you can use a default model if you provide a language pair in the task name (such as "translation_en_to_fr"), but the easiest way is to pick the model you want to use on the Model Hub. Here we'll try translating from French to English:

```
from transformers import pipeline

translator = pipeline("translation", model="Helsinki-NLP/opus-mt-fr-en")
translator("Ce cours est produit par Hugging Face.")
```

```
[{'translation_text': 'This course is produced by Hugging Face.'}]
```

Like with text generation and summarization, you can specify a max_length or a min_length for the result.

> ✏️ **Try it out!** Search for translation models in other languages and try to translate the previous sentence into a few different languages.

Playing with Transformers (V)

Translation

For translation, you can use a default model if you provide a language pair in the task name (such as "translation_en_to_fr"), but the easiest way is to pick the model you want to use on the Model Hub. Here we'll try translating from French to English:

```
from transformers import pipeline

translator = pipeline("translation", model="Helsinki-NLP/opus-mt-fr-en")
translator("Ce cours est produit par Hugging Face.")
```

```
[{'translation_text': 'This course is produced by Hugging Face.'}]
```

Like with text generation and summarization, you can specify a max_length or a min_length for the result.

> ✏️ **Try it out!** Search for translation models in other languages and try to translate the previous sentence into a few different languages.

First Chapter Quiz

End-of-chapter quiz - Hu | Models (PyTorch) - Colab | hugging face checkpoint | WhatsApp

https://huggingface.co/learn/nlp-course/chapter1/10?fw=pt

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

NLP Course

Search documentation Ctrl+K

EN | 2,035

End-of-chapter quiz

Ask a question

This chapter covered a lot of ground! Don't worry if you didn't grasp all the details; the next chapters will help you understand how things work under the hood.

First, though, let's test what you learned in this chapter!

1. Explore the Hub and look for the roberta-large-mnli checkpoint. What task does it perform?

- [ ] Summarization
- [ ] Text classification
- [ ] Text generation

Submit

2. What will the following code return?

```
from transformers import pipeline

ner = pipeline("ner", grouped_entities=True)
ner("My name is Sylvain and I work at Hugging Face in Brooklyn.")
```

- [ ] It will return classification scores for this sentence, with labels "positive" or "negative".
- [ ] It will return a generated text completing this sentence.
- [ ] It will return the words representing persons, organizations or locations.

Submit

Behind the pipeline

Behind the pipeline - Hu | Models (PyTorch) - Colab | Berga Resort | de 41.4890908, 2.04226 | hugging face checkpoin | WhatsApp

huggingface.co/learn/nlp-course/chapter2/2?fw=pt

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

NLP Course

Search documentation Ctrl+K

EN | 2,035

Pytorch | TensorFlow

Ask a question | Open in Colab | Open Studio Lab

Behind the pipeline

> This is the first section where the content is slightly different depending on whether you use PyTorch or TensorFlow. Toggle the switch on top of the title to select the platform you prefer!

Let's start with a complete example, taking a look at what happened behind the scenes when we executed the following code in Chapter 1:

```
from transformers import pipeline
```

Second Chapter Quiz