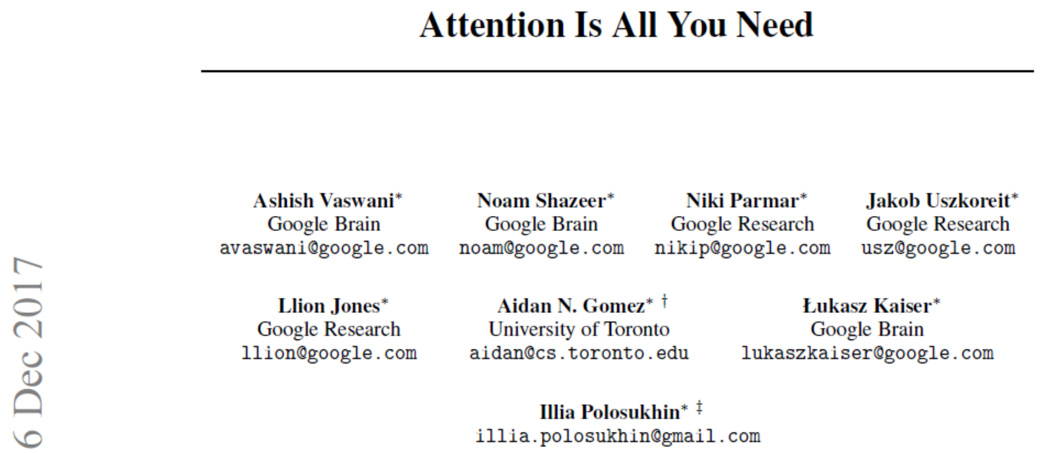
Data Analysis

Practice 8: Transformers

Dr. Nataliya K. Sakhnenko

The Transformer



6 Dec 2017

Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

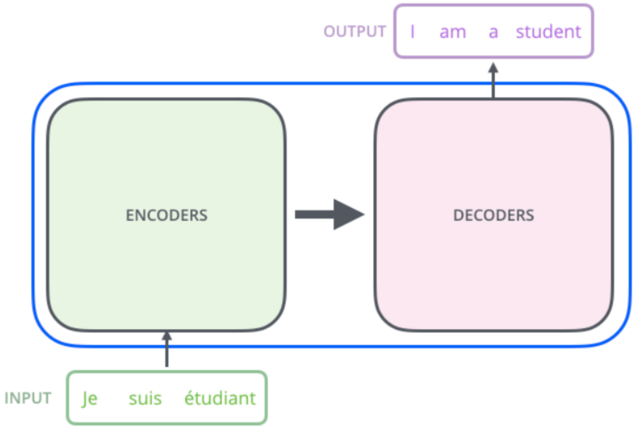
Jakob Uszkoreit*
Google Research
usz@google.com

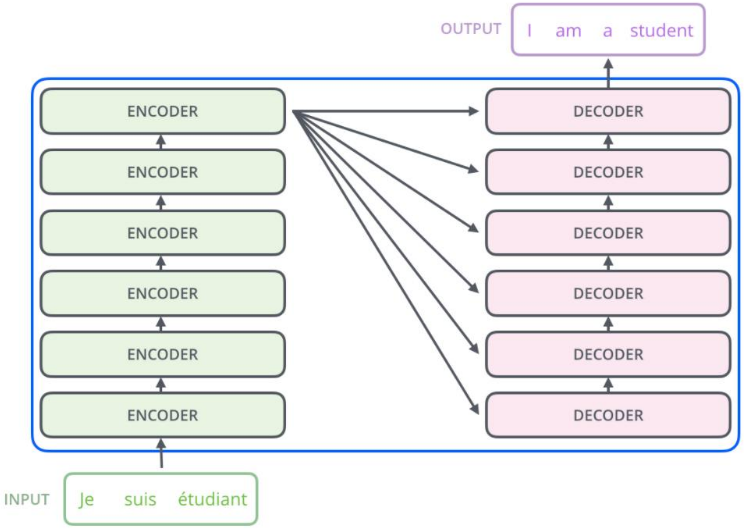
Llion Jones*
Google Research
llion@google.com

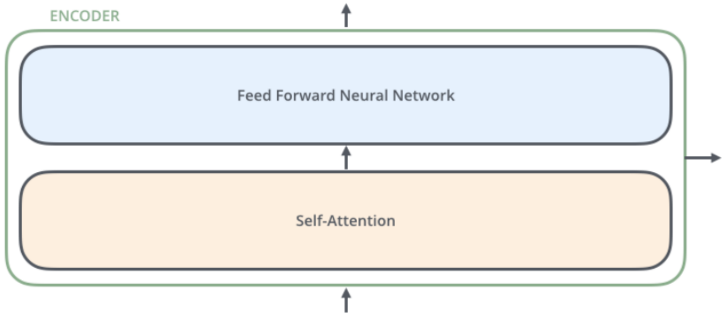
Aidan N. Gomez* †
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser*
Google Brain
lukaszkaiser@google.com

Illia Polosukhin* ‡
illia.polosukhin@gmail.com







Examples

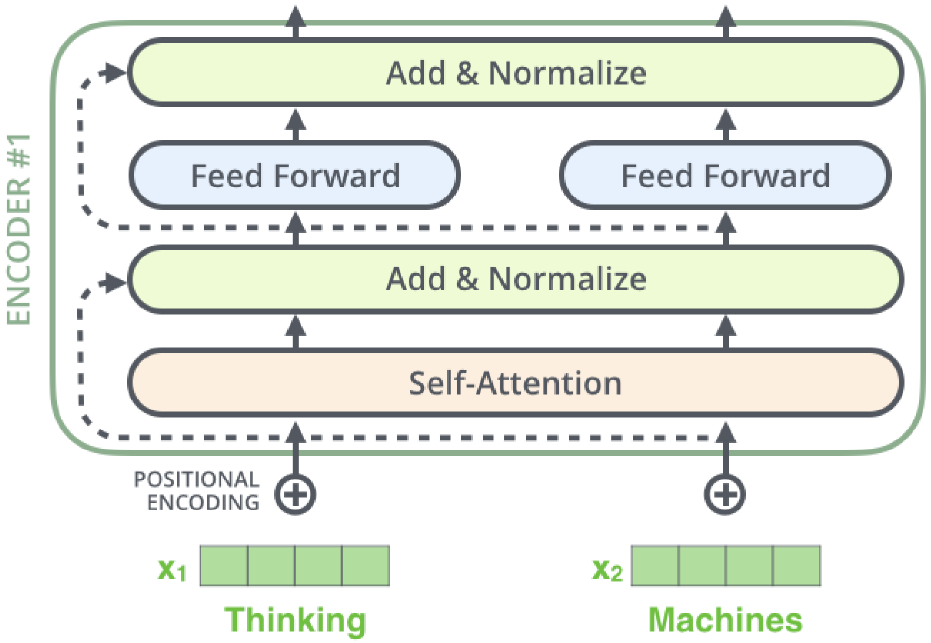
- Text classification with transformers

https://keras.io/examples/nlp/text_classification_with_transformer/

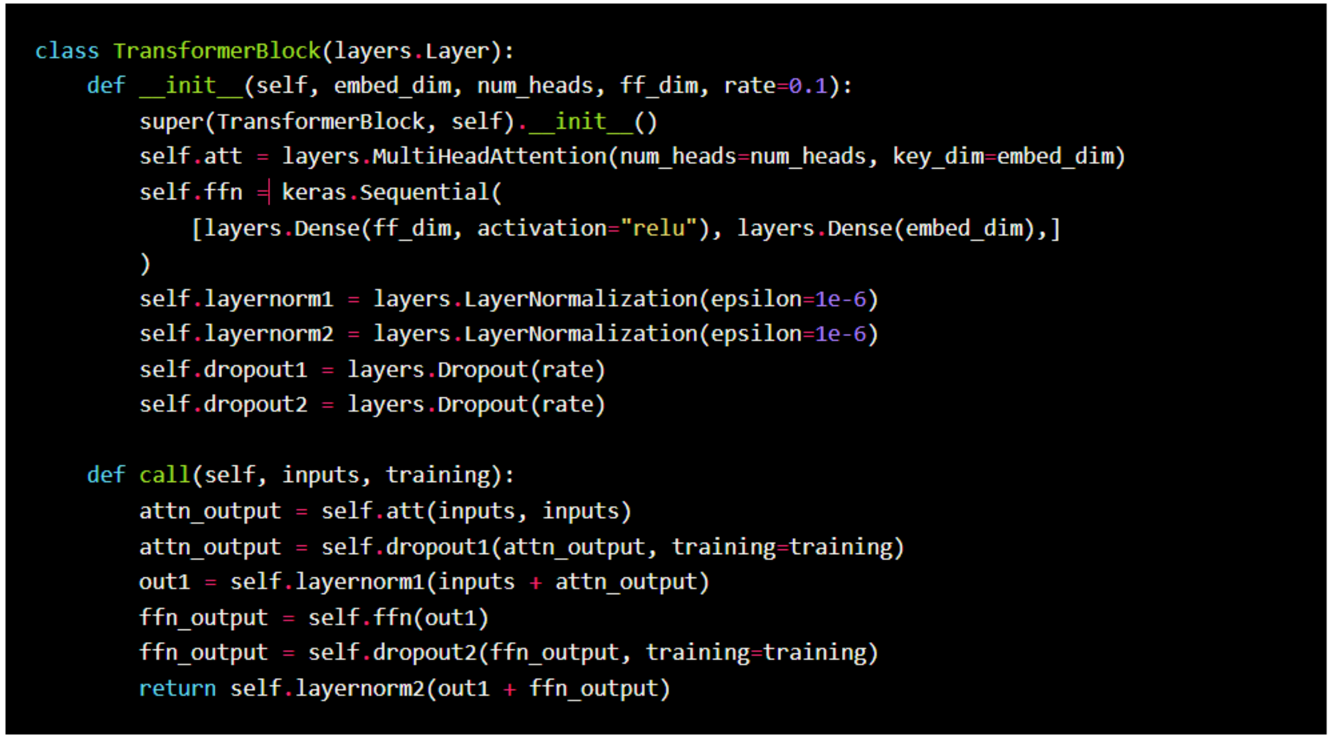
- English-to-Spanish translation with a sequence-to-sequence Transformer

https://keras.io/examples/nlp/neural_machine_translation_with_transformer/

Encoder implementation



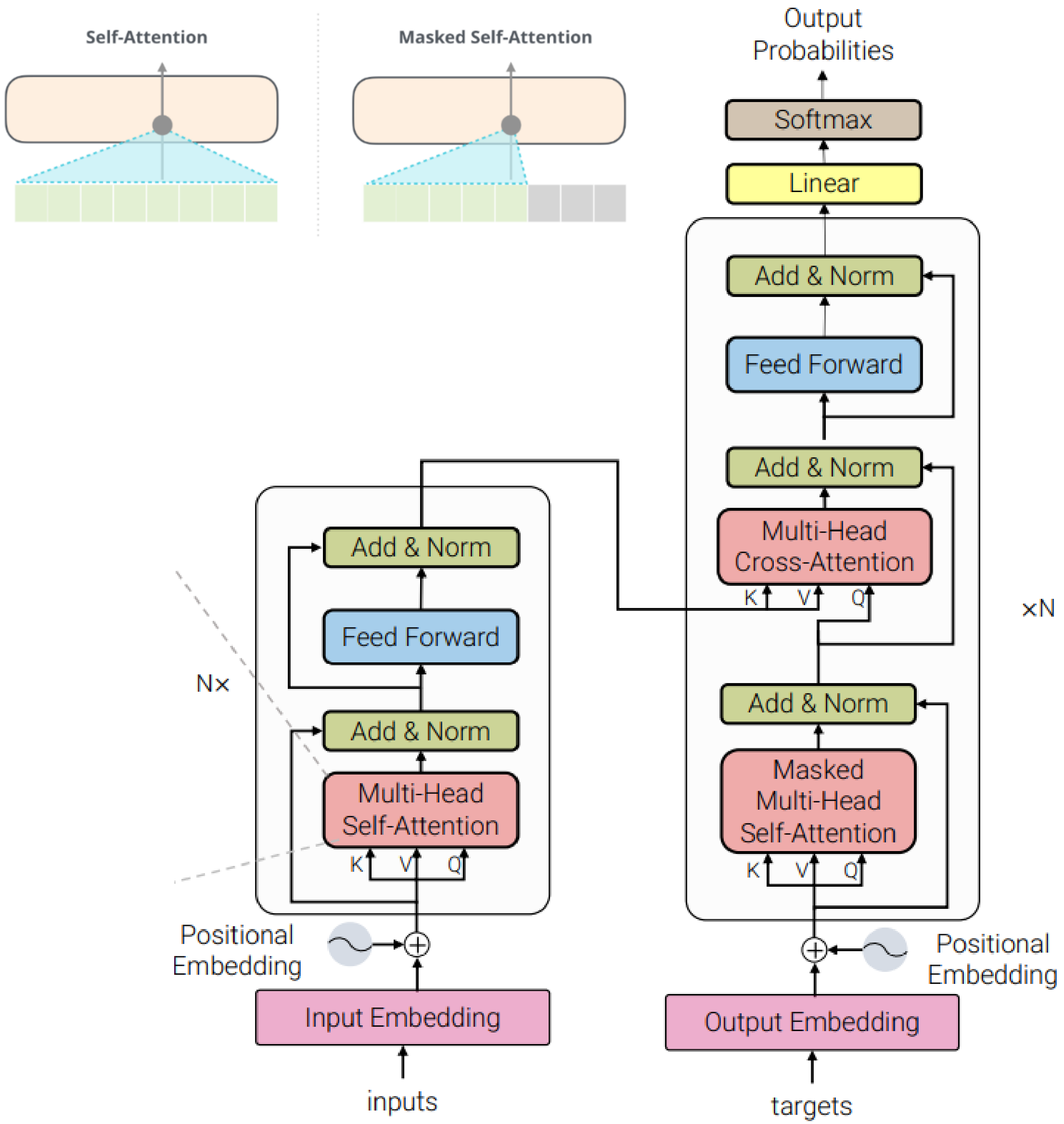
Implement a Transformer block as a layer

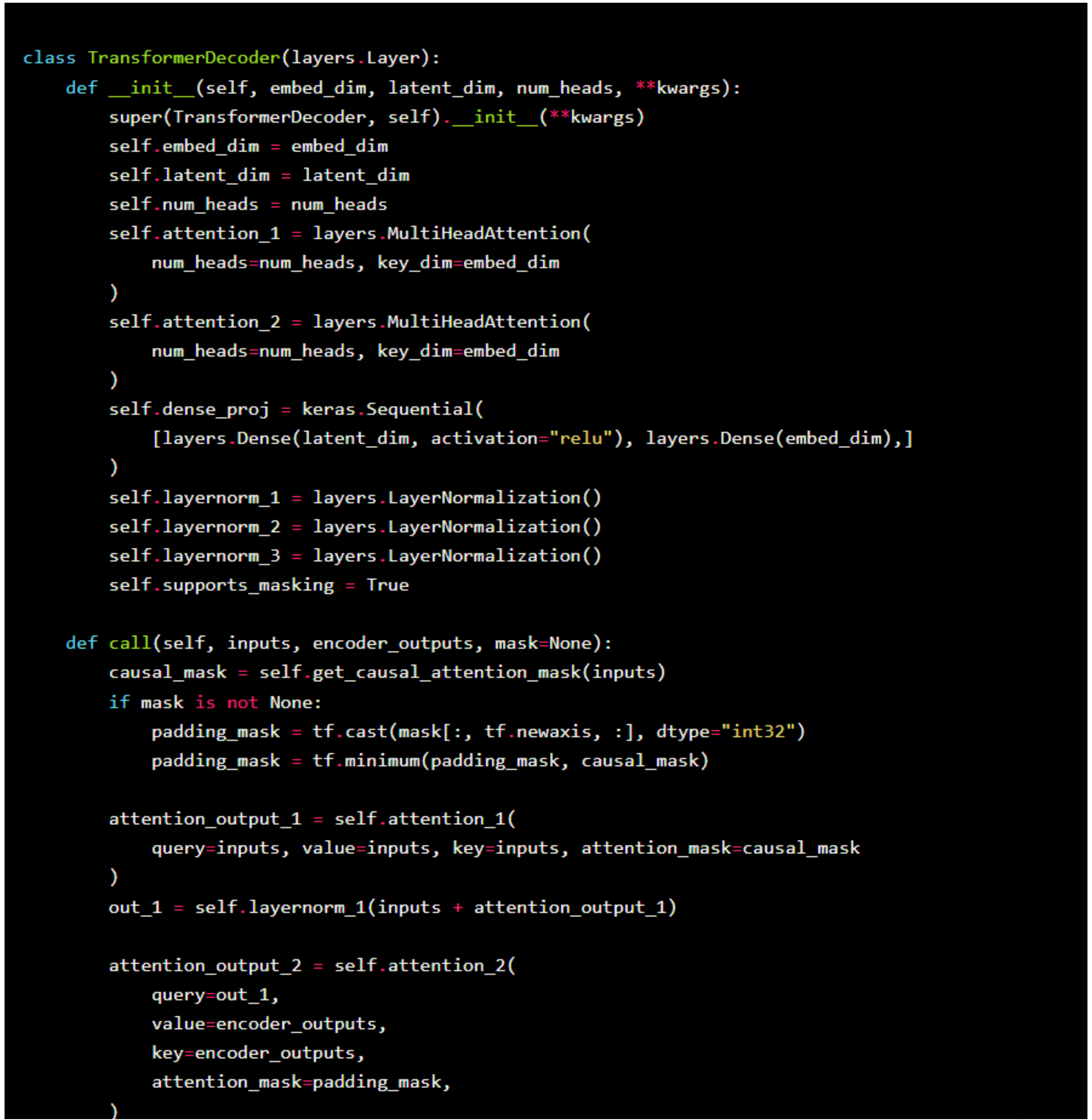


```
class TransformerBlock(layers.Layer):
    def __init__(self, embed_dim, num_heads, ff_dim, rate=0.1):
        super(TransformerBlock, self).__init__()
        self.att = layers.MultiHeadAttention(num_heads=num_heads, key_dim=embed_dim)
        self.ffn = keras.Sequential(
            [layers.Dense(ff_dim, activation="relu"), layers.Dense(embed_dim),]
        )
        self.layernorm1 = layers.LayerNormalization(epsilon=1e-6)
        self.layernorm2 = layers.LayerNormalization(epsilon=1e-6)
        self.dropout1 = layers.Dropout(rate)
        self.dropout2 = layers.Dropout(rate)

    def call(self, inputs, training):
        attn_output = self.att(inputs, inputs)
        attn_output = self.dropout1(attn_output, training=training)
        out1 = self.layernorm1(inputs + attn_output)
        ffn_output = self.ffn(out1)
        ffn_output = self.dropout2(ffn_output, training=training)
        return self.layernorm2(out1 + ffn_output)
```

https://keras.io/examples/nlp/text_classification_with_transformer/

Decoder implementation



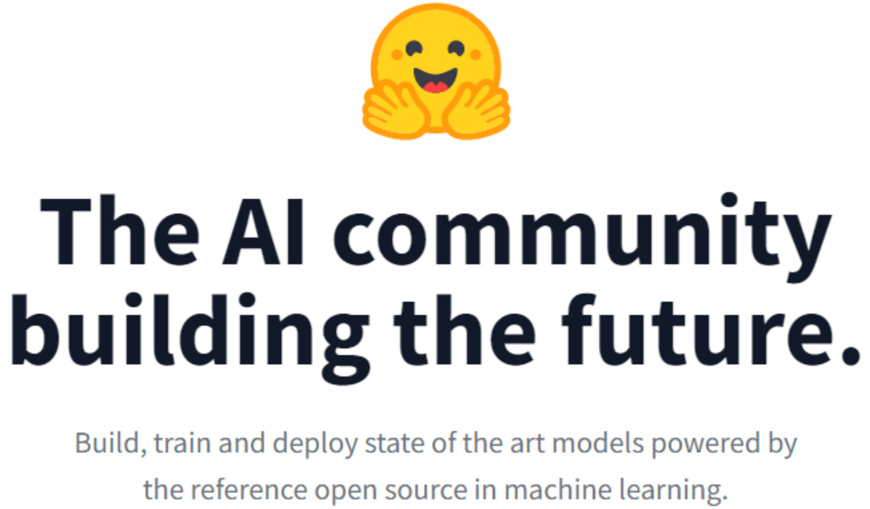


```
class TransformerDecoder(layers.Layer):
    def __init__(self, embed_dim, latent_dim, num_heads, **kwargs):
        super(TransformerDecoder, self).__init__(**kwargs)
        self.embed_dim = embed_dim
        self.latent_dim = latent_dim
        self.num_heads = num_heads
        self.attention_1 = layers.MultiHeadAttention(
            num_heads=num_heads, key_dim=embed_dim
        )
        self.attention_2 = layers.MultiHeadAttention(
            num_heads=num_heads, key_dim=embed_dim
        )
        self.dense_proj = keras.Sequential(
            [layers.Dense(latent_dim, activation="relu"), layers.Dense(embed_dim),]
        )
        self.layernorm_1 = layers.LayerNormalization()
        self.layernorm_2 = layers.LayerNormalization()
        self.layernorm_3 = layers.LayerNormalization()
        self.supports_masking = True

    def call(self, inputs, encoder_outputs, mask=None):
        causal_mask = self.get_causal_attention_mask(inputs)
        if mask is not None:
            padding_mask = tf.cast(mask[:, tf.newaxis, :], dtype="int32")
            padding_mask = tf.minimum(padding_mask, causal_mask)

        attention_output_1 = self.attention_1(
            query=inputs, value=inputs, key=inputs, attention_mask=causal_mask
        )
        out_1 = self.layernorm_1(inputs + attention_output_1)

        attention_output_2 = self.attention_2(
            query=out_1,
            value=encoder_outputs,
            key=encoder_outputs,
            attention_mask=padding_mask,
        )
```

https://keras.io/examples/nlp/neural_machine_translation_with_transformer/

🤗 **Hugging Face**

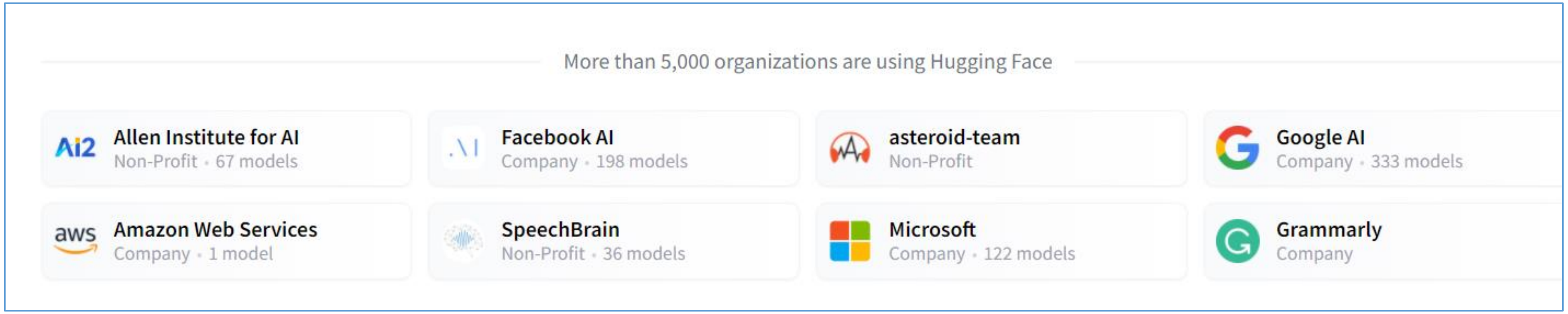
Hugging Face ecosystem:
🤗 Transformers, 🤗 Datasets, 🤗 Tokenizers, and 🤗 Accelerate



The AI community building the future.

Build, train and deploy state of the art models powered by the reference open source in machine learning.

Hugging Face Transformers



More than 5,000 organizations are using Hugging Face

Allen Institute for AI
Non-Profit · 67 models

Facebook AI
Company · 198 models

asteroid-team
Non-Profit

Google AI
Company · 333 models

Amazon Web Services
Company · 1 model

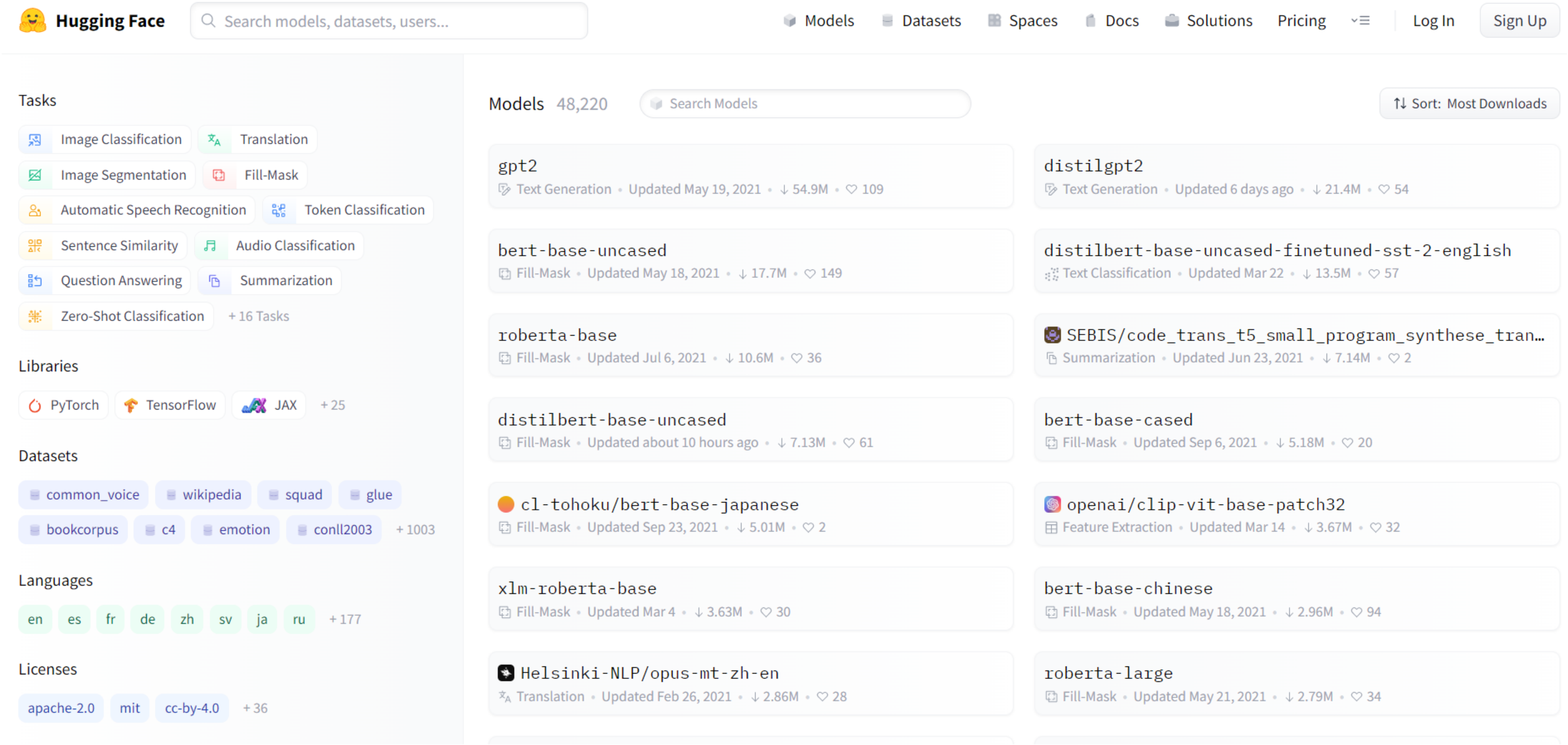
SpeechBrain
Non-Profit · 36 models

Microsoft
Company · 122 models

Grammarly
Company

https://huggingface.co/

Hugging Face Transformers



Tasks

Image Classification | Translation | Image Segmentation | Fill-Mask | Automatic Speech Recognition | Token Classification | Sentence Similarity | Audio Classification | Question Answering | Summarization | Zero-Shot Classification | + 16 Tasks

Libraries

PyTorch | TensorFlow | JAX | + 25

Datasets

common_voice | wikipedia | squad | glue | bookcorpus | c4 | emotion | conll2003 | + 1003

Languages

en | es | fr | de | zh | sv | ja | ru | + 177

Licenses

apache-2.0 | mit | cc-by-4.0 | + 36

Models 48,220 | Search Models | Sort: Most Downloads

gpt2
Text Generation • Updated May 19, 2021 • ↓ 54.9M • ♡ 109

distilgpt2
Text Generation • Updated 6 days ago • ↓ 21.4M • ♡ 54

bert-base-uncased
Fill-Mask • Updated May 18, 2021 • ↓ 17.7M • ♡ 149

distilbert-base-uncased-finetuned-sst-2-english
Text Classification • Updated Mar 22 • ↓ 13.5M • ♡ 57

roberta-base
Fill-Mask • Updated Jul 6, 2021 • ↓ 10.6M • ♡ 36

SEBIS/code_trans_t5_small_program_synthese_tran…
Summarization • Updated Jun 23, 2021 • ↓ 7.14M • ♡ 2

distilbert-base-uncased
Fill-Mask • Updated about 10 hours ago • ↓ 7.13M • ♡ 61

bert-base-cased
Fill-Mask • Updated Sep 6, 2021 • ↓ 5.18M • ♡ 20

cl-tohoku/bert-base-japanese
Fill-Mask • Updated Sep 23, 2021 • ↓ 5.01M • ♡ 2

openai/clip-vit-base-patch32
Feature Extraction • Updated Mar 14 • ↓ 3.67M • ♡ 32

xlm-roberta-base
Fill-Mask • Updated Mar 4 • ↓ 3.63M • ♡ 30

bert-base-chinese
Fill-Mask • Updated May 18, 2021 • ↓ 2.96M • ♡ 94

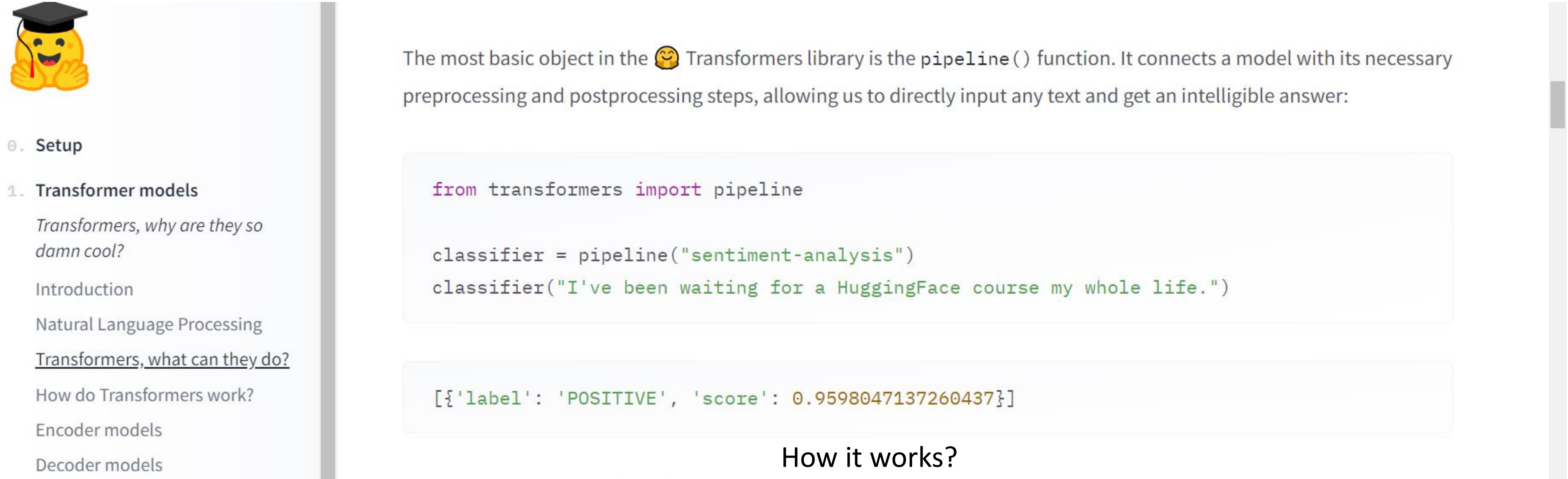
Helsinki-NLP/opus-mt-zh-en
Translation • Updated Feb 26, 2021 • ↓ 2.86M • ♡ 28

roberta-large
Fill-Mask • Updated May 21, 2021 • ↓ 2.79M • ♡ 34

https://huggingface.co/models

HF Transformers: working with the pipeline

https://huggingface.co/course/chapter1/



0. Setup

1. Transformer models

   Transformers, why are they so damn cool?

   Introduction

   Natural Language Processing

   Transformers, what can they do?

   How do Transformers work?

   Encoder models

   Decoder models

   Sequence-to-sequence models

   Bias and limitations

   Summary

   End-of-chapter quiz

2. Using 🤗 Transformers

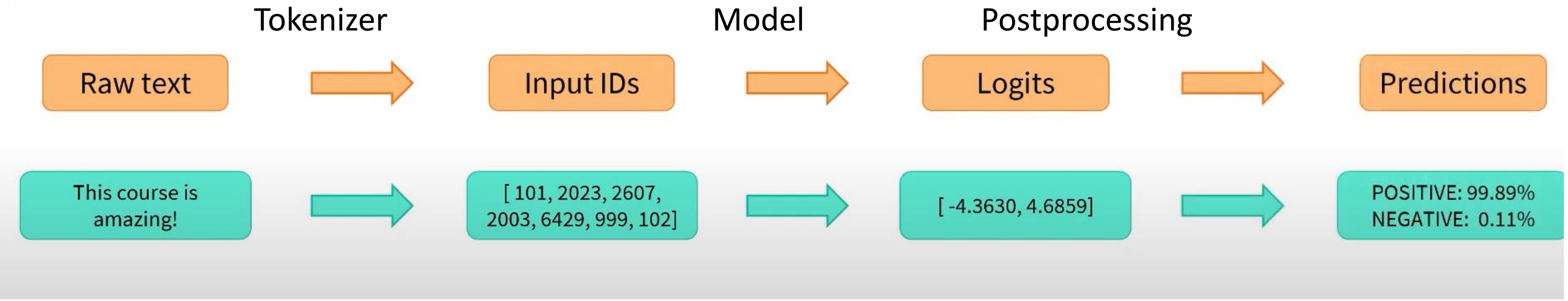
3. Fine-tuning a pretrained model

The most basic object in the 🤗 Transformers library is the `pipeline()` function. It connects a model with its necessary preprocessing and postprocessing steps, allowing us to directly input any text and get an intelligible answer:

```
from transformers import pipeline

classifier = pipeline("sentiment-analysis")
classifier("I've been waiting for a HuggingFace course my whole life.")
```

```
[{'label': 'POSITIVE', 'score': 0.9598047137260437}]
```

How it works?



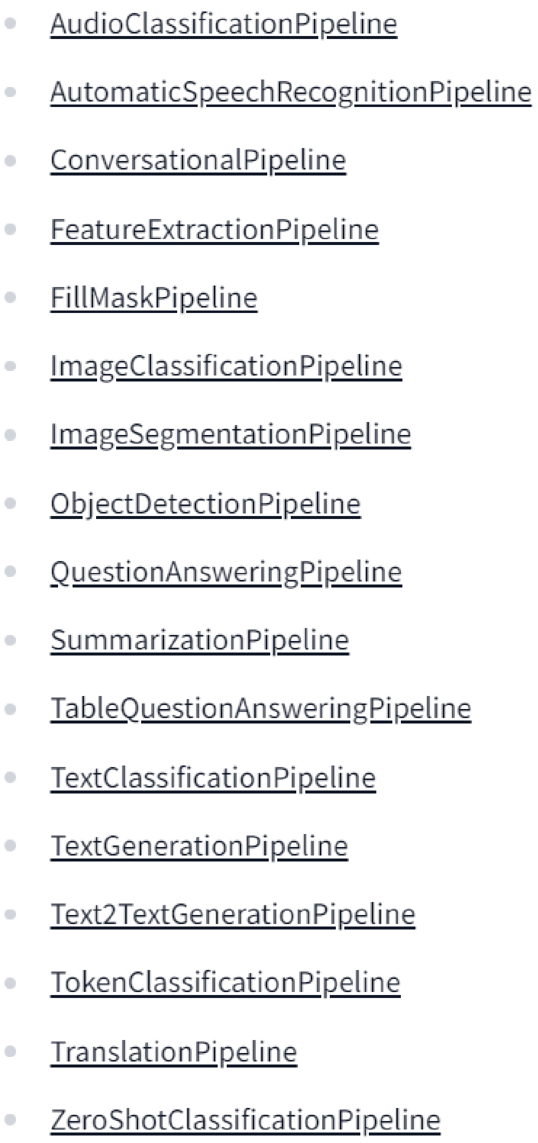
Tokenizer → Model → Postprocessing

Raw text → Input IDs → Logits → Predictions

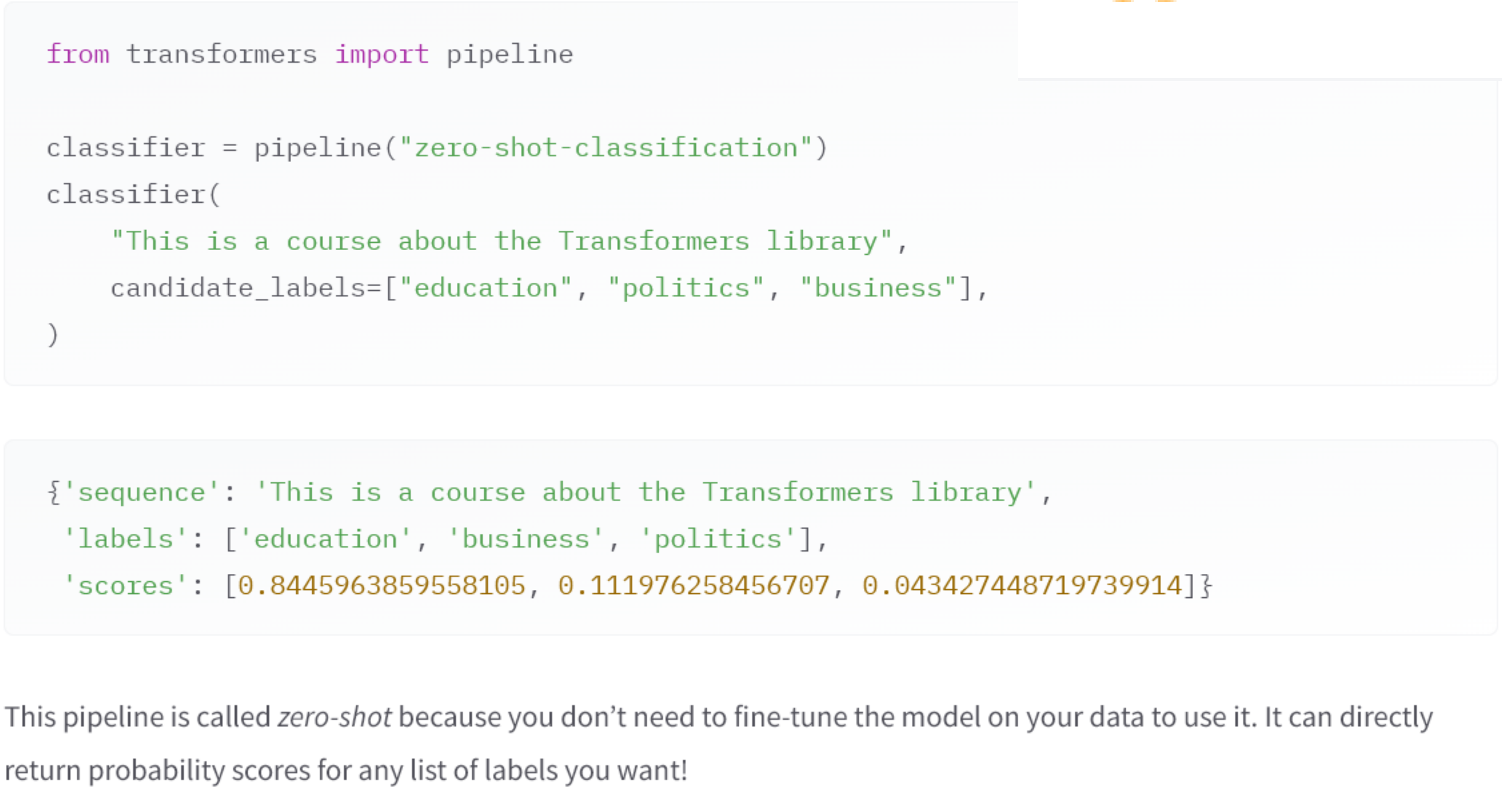
This course is amazing! → [101, 2023, 2607, 2003, 6429, 999, 102] → [-4.3630, 4.6859] → POSITIVE: 99.89% NEGATIVE: 0.11%

HF: Pipelines

https://huggingface.co/docs/transformers/main_classes/pipelines



- AudioClassificationPipeline
- AutomaticSpeechRecognitionPipeline
- ConversationalPipeline
- FeatureExtractionPipeline
- FillMaskPipeline
- ImageClassificationPipeline
- ImageSegmentationPipeline
- ObjectDetectionPipeline
- QuestionAnsweringPipeline
- SummarizationPipeline
- TableQuestionAnsweringPipeline
- TextClassificationPipeline
- TextGenerationPipeline
- Text2TextGenerationPipeline
- TokenClassificationPipeline
- TranslationPipeline
- ZeroShotClassificationPipeline

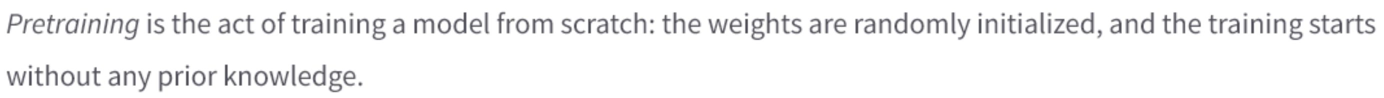


```
from transformers import pipeline

classifier = pipeline("zero-shot-classification")
classifier(
    "This is a course about the Transformers library",
    candidate_labels=["education", "politics", "business"],
)
```

```
{'sequence': 'This is a course about the Transformers library',
 'labels': ['education', 'business', 'politics'],
 'scores': [0.8445963859558105, 0.111976258456707, 0.043427448719739914]}
```

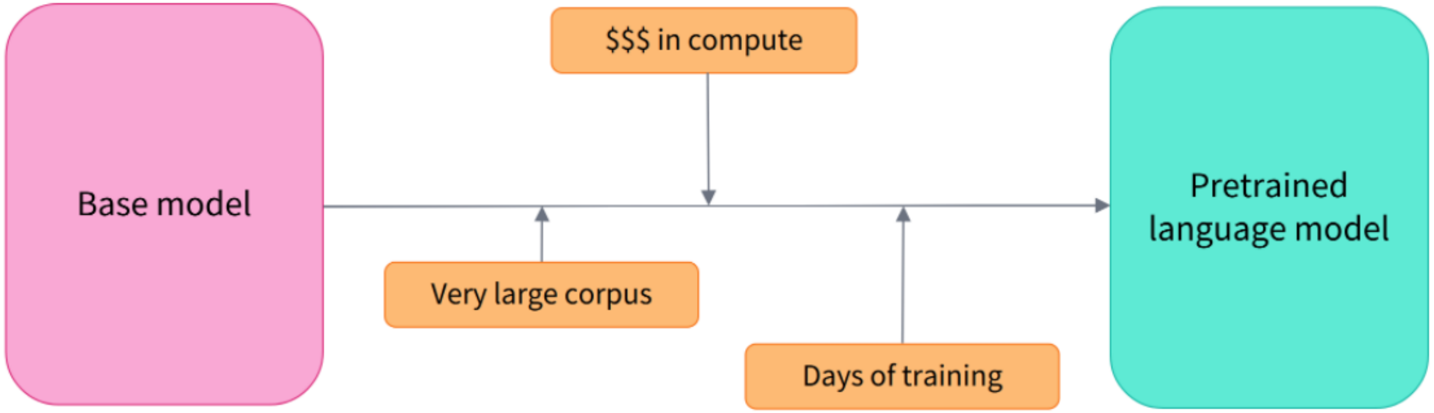
This pipeline is called *zero-shot* because you don’t need to fine-tune the model on your data to use it. It can directly return probability scores for any list of labels you want!

HF: Fine-tuning a model with the Trainer API



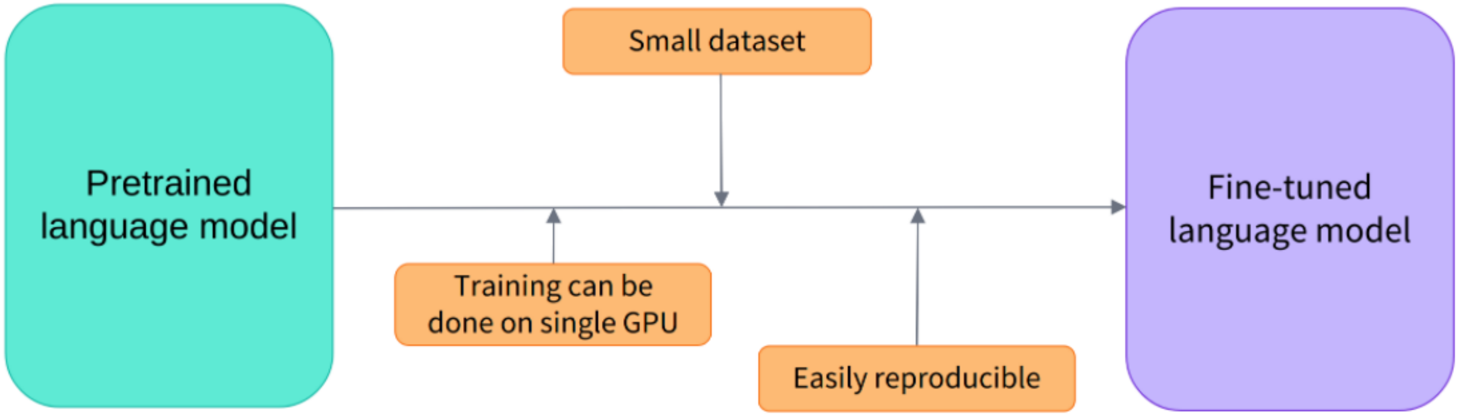
Pretraining is the act of training a model from scratch: the weights are randomly initialized, and the training starts without any prior knowledge.

https://huggingface.co/course/chapter1/



```
from transformers import Trainer

trainer = Trainer(model=model,
            args=training_args,
            train_dataset=small_train_dataset,
            eval_dataset=small_eval_dataset)
```

https://huggingface.co/docs/transformers/training



Pretrained language model
Small dataset
Training can be done on single GPU
Easily reproducible
Fine-tuned language model