Open source AI with Hugging Face

Julien Simon, Chief Evangelist, Hugging Face
julsimon@huggingface.co
youtube.com/juliensimonfr

June 2017: The Transformer architecture

https://arxiv.org/abs/1706.03762

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

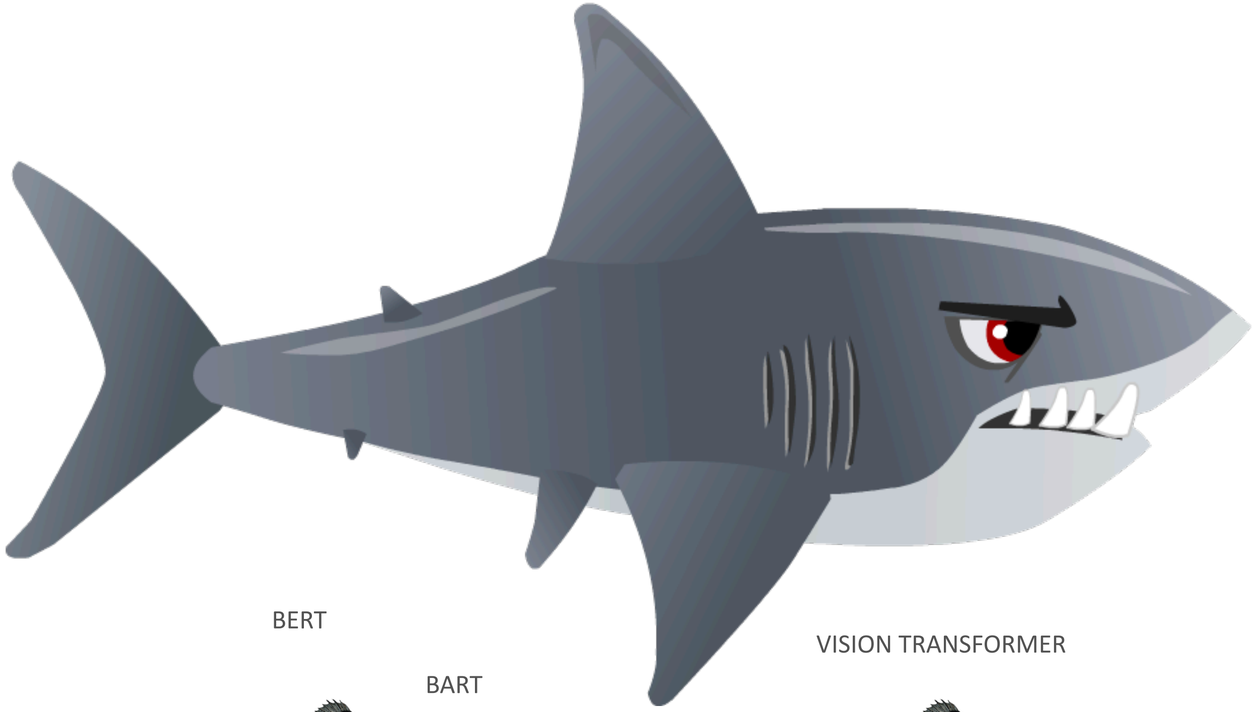
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

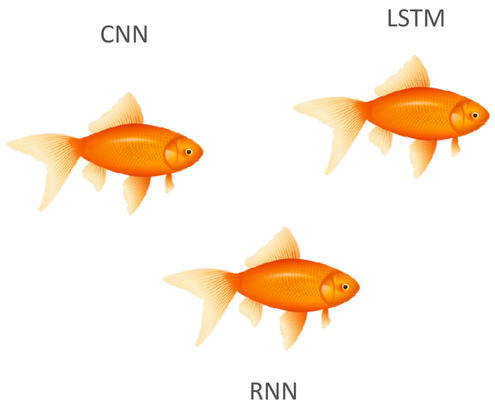
Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

"We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely."

2022: Transformers are eating Deep Learning

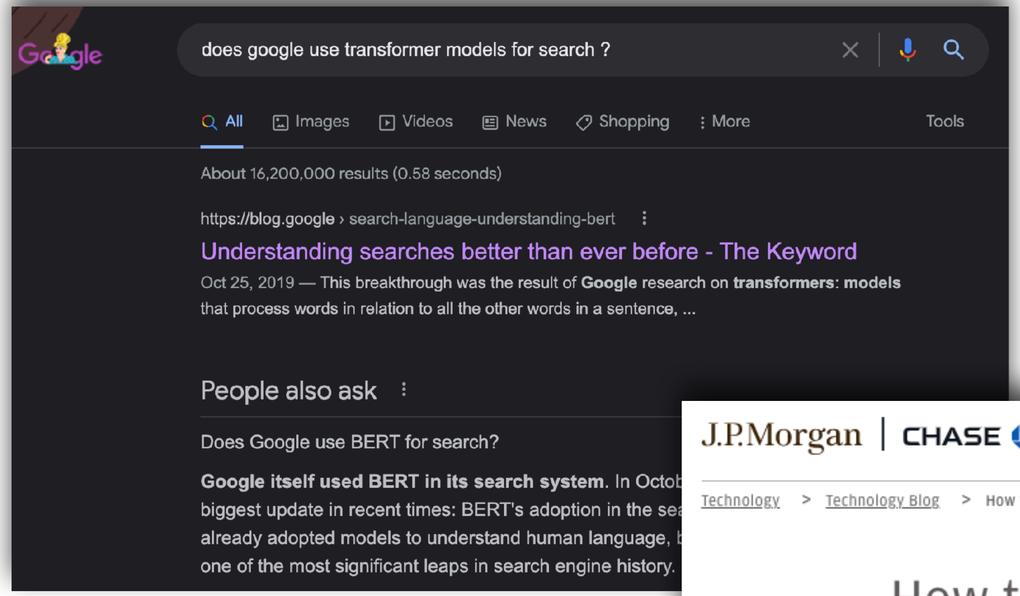




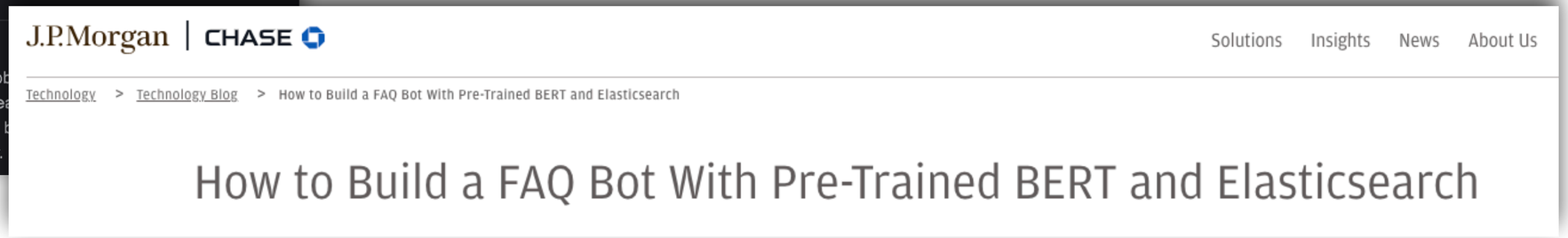
"*Transformers are emerging as a general-purpose architecture for ML*"
https://www.stateof.ai (2021)

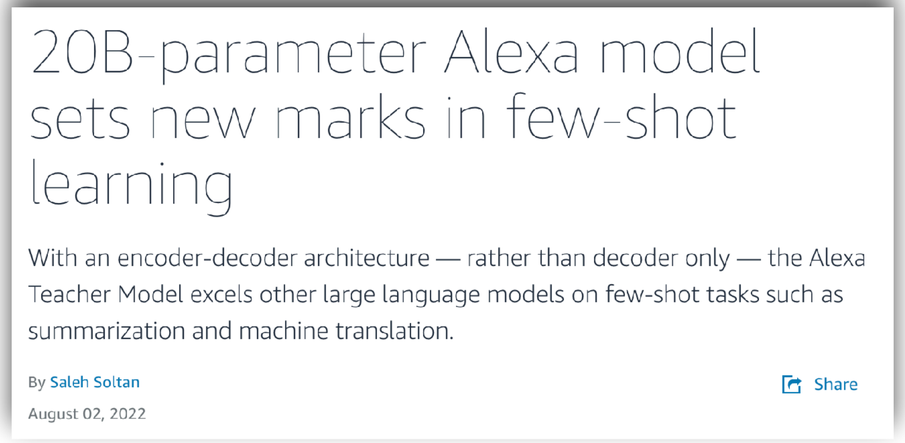
RNN and CNN usage down, Transformers usage up!
https://www.kaggle.com/kaggle-survey-2021

Transformer models in the wild



Under the Hood at Grammarly: Leveraging Transformer Language Models for Grammatical Error Correction





Generative models in the wild

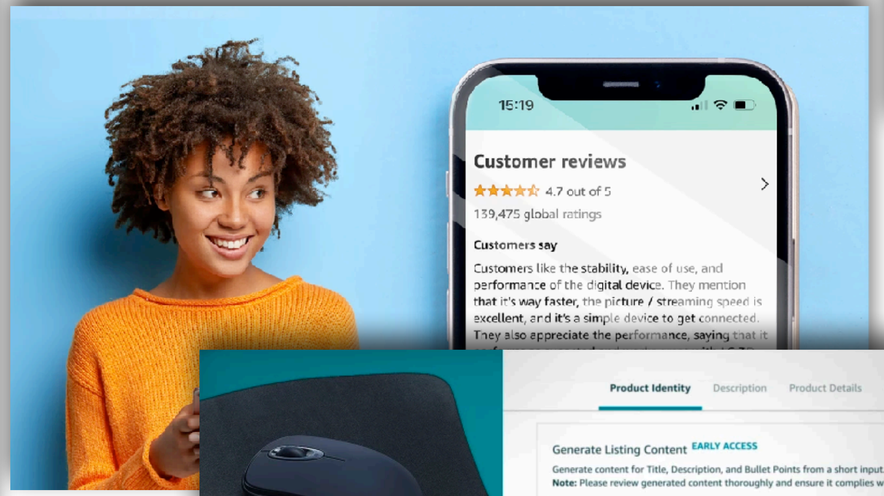
JU

is ChatGPT powered by transformer models?

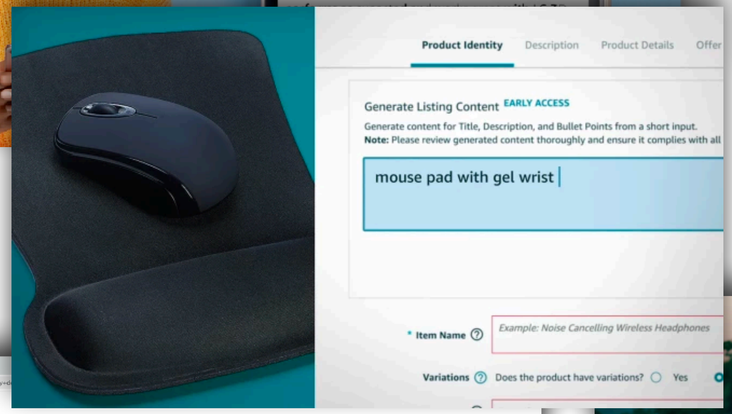
Yes, ChatGPT is powered by transformer models. Specifically, it is based on the architecture of the GPT (Generative Pre-trained Transformer) family of models, which were developed by OpenAI. The GPT models are designed to generate natural language text, and they use a transformer architecture that allows them to model long-term dependencies in language.

ChatGPT is a variant of the GPT models that has been fine-tuned on a large amount of conversational data in order to generate human-like responses to user input. It uses a combination of neural network architecture, training data, and machine learning algorithms to generate responses that are contextually appropriate and semantically relevant.

GitHub Copilot

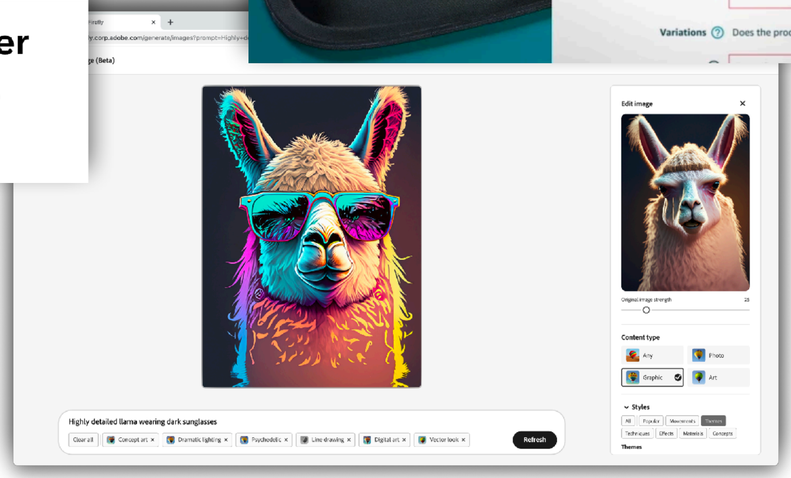


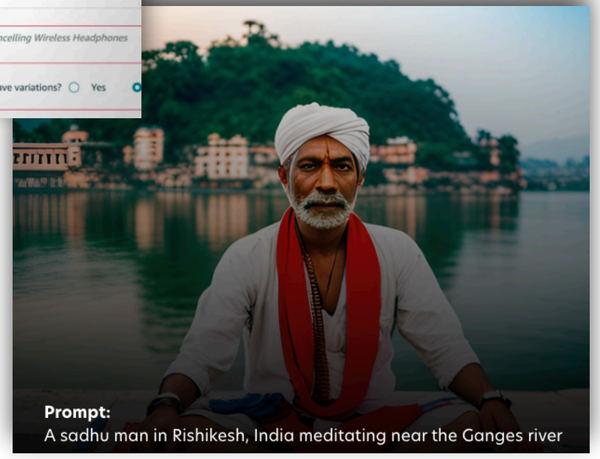
Amazon CodeWhisperer



Introducing BloombergGPT, Bloomberg's 50-billion parameter large language model, purpose-built from scratch for finance

March 30, 2023



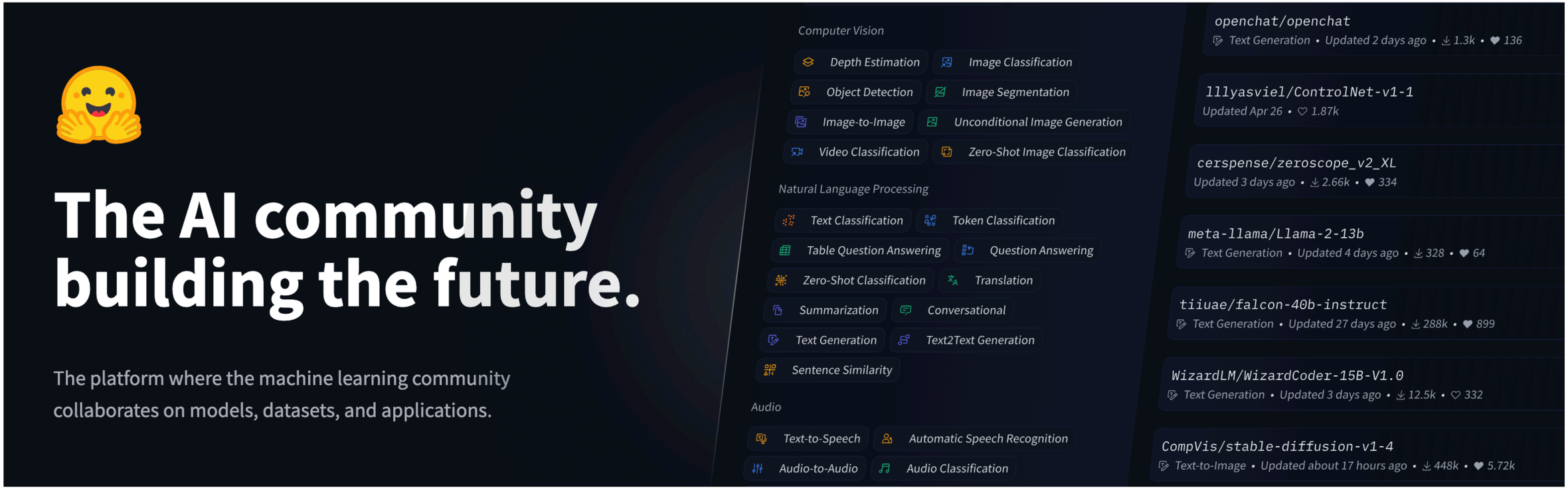


Meet Einstein GPT, the World's First Generative AI for CRM

LEARN MORE →

Hugging Face: the focal point of open-source AI

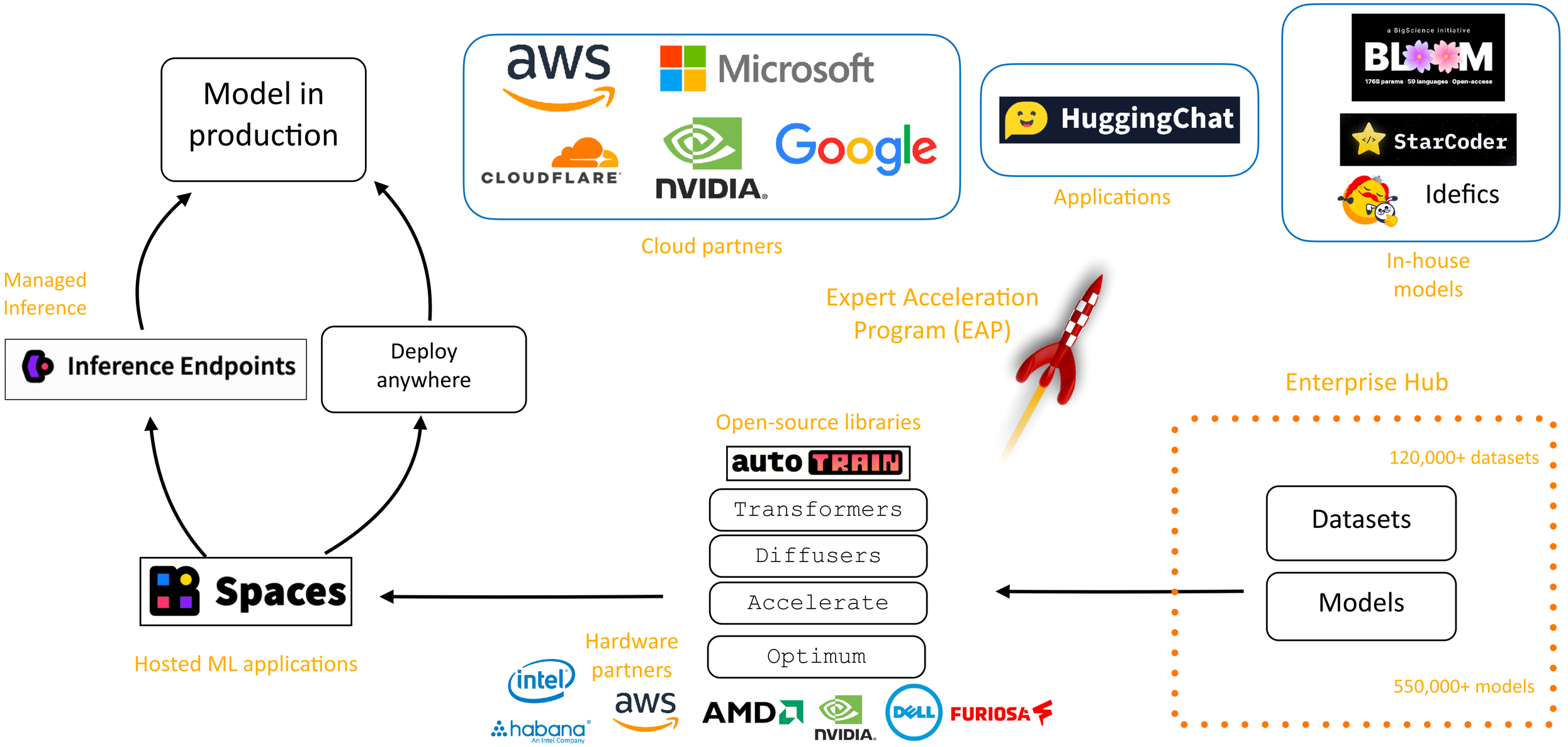
https://huggingface.co



Backed by the largest cloud and chip companies

salesforce aws Google IBM Microsoft AMD intel NVIDIA QUALCOMM

Hugging Face at a glance



Example: zero-shot classification

```python
from transformers import pipeline

classifier = pipeline("zero-shot-classification", model="facebook/bart-large-mnli")

sentence = "The Black–Scholes /ˌblæk ˈʃoʊlz/[1] or Black–Scholes–Merton model is a mathematical model
for the dynamics of a financial market containing derivative investment instruments. From the
parabolic partial differential equation in the model, known as the Black–Scholes equation, one can
deduce the Black–Scholes formula, which gives a theoretical estimate of the price of European-style
options and shows that the option has a unique price given the risk of the security and its expected
return (instead replacing the security's expected return with the risk-neutral rate). The equation
and model are named after economists Fischer Black and Myron Scholes."

candidate_labels = ['finance', 'science', 'business', 'stock market', 'research']

result = classifier(sentence, candidate_labels)

# 'labels': ['finance', 'science', 'research', 'business', 'stock market']
# 'scores': [0.6812, 0.1999, 0.0745, 0.0268 0.0173]
```

Start learning at huggingface.co/tasks and huggingface.co/course

Example: text-to-image generation

https://huggingface.co/spaces/raannakasturi/SDXL

Stable Diffusion XL v0.9, June 2023

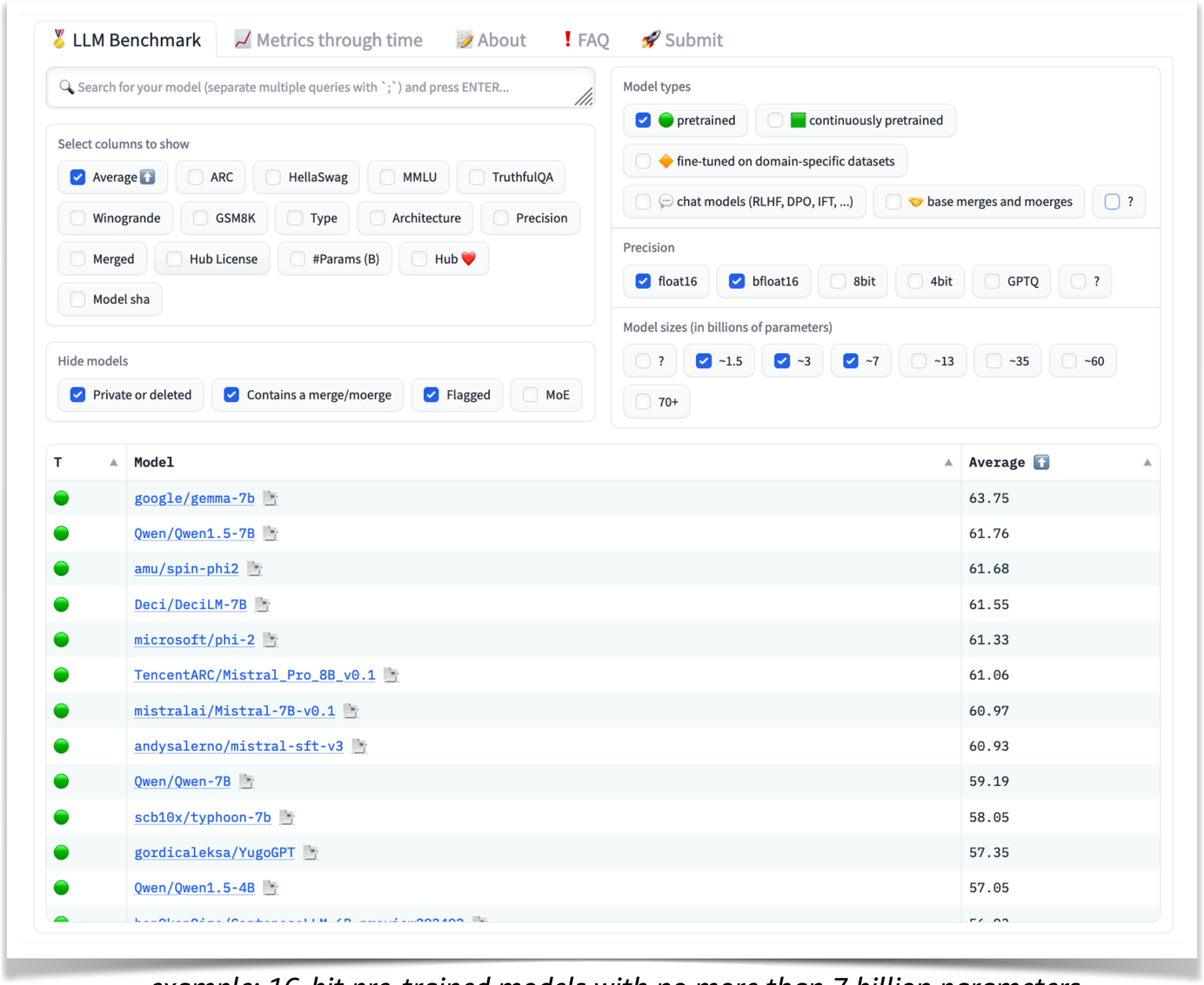
```
from diffusers import StableDiffusionPipeline

pipe = StableDiffusionPipeline.from_pretrained(
  "stabilityai/stable-diffusion-xl-base-0.9")

prompt = "in the desert, a Corvette parked \
in front of an old-school diner at sundown"

images = pipe(prompt)
images[0].save("picture.png")
```

Why customers are more successful with open-source models

- **Accessibility**: you can use models regardless of budget or affiliation
- **Transparency**: you have full visibility on model architecture and weights
- **Privacy**: you don't have to send their data to black box APIs
- **IP protection**: you train models on your data, and own them
- **Freedom of choice**: you are not locked in. You can switch models anytime
- **IT flexibility**: you can train and deploy models anywhere you like, using any technology
- **Cost optimization**: you can find the cost/performance sweet spot for each project
- **Model quality**: a small fine-tuned model will almost always outperform a generic large model

Picking models with the Open LLM leaderboard (1/2)

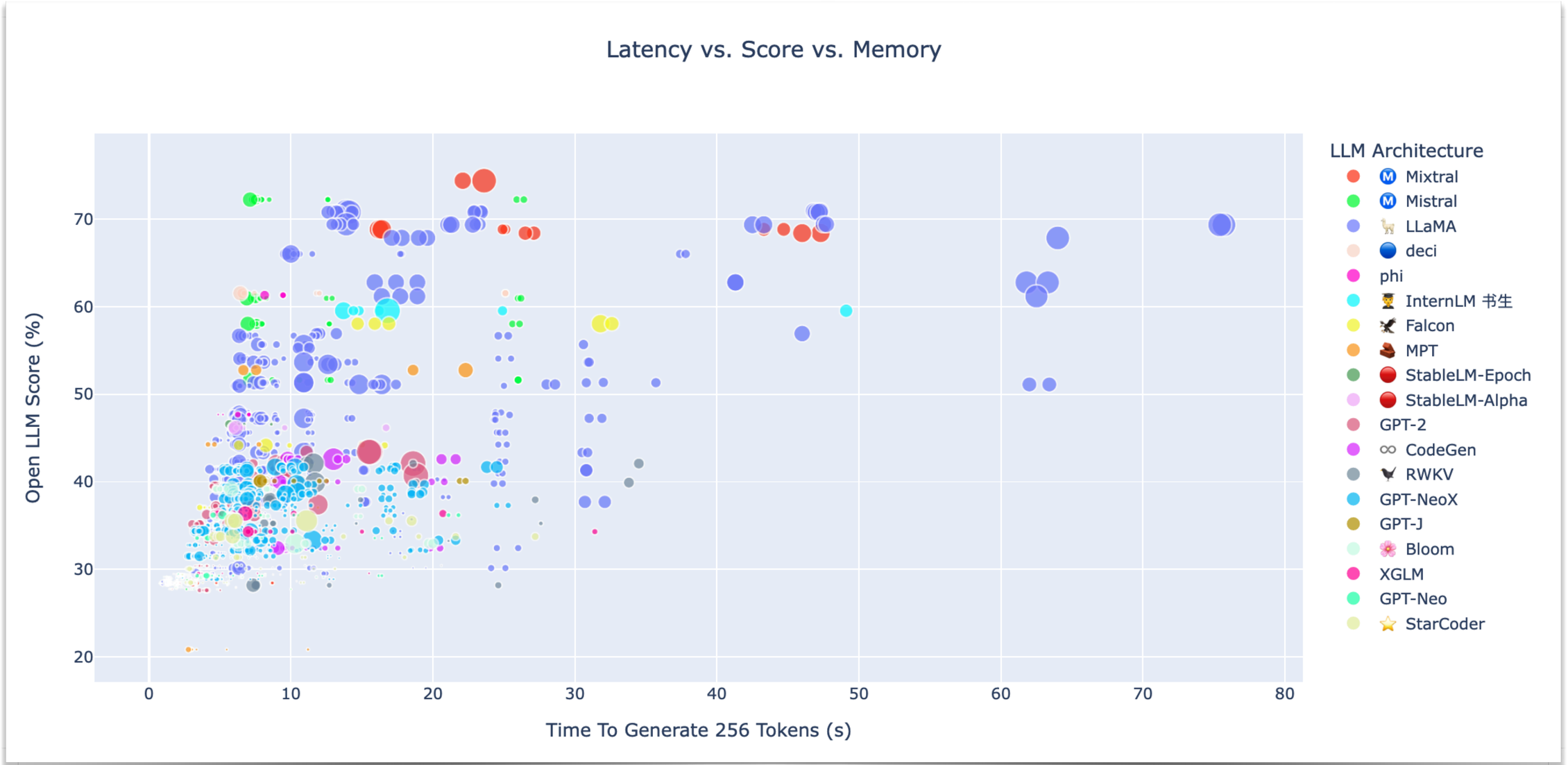
https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard



example: 16-bit pre-trained models with no more than 7 billion parameters

Picking models with the Open LLM leaderboard (2/2)

https://huggingface.co/spaces/optimum/llm-perf-leaderboard



Picking embeddings with the MTEB leaderboard

https://huggingface.co/spaces/mteb/leaderboard

Massive Text Embedding Benchmark (MTEB) Leaderboard. To submit, refer to the MTEB GitHub repository 🤗 Refer to the MTEB paper for details on metrics, tasks and models.

Overall | Bitext Mining | Classification | Clustering | Pair Classification | Reranking | Retrieval | STS | Summarization

English | Chinese | French | Polish

Overall MTEB English leaderboard 🔮

- **Metric:** Various, refer to task tabs
- **Languages:** English

Rank	Model	Model Size (GB)	Embedding Dimensions	Max Tokens	Average (56 datasets)	Classification Average (12 datasets)	Clustering Average (11 datasets)	Pair Classification Average (3 datasets)
1	SFR-Embedding-Mistral	14.22	4096	32768	67.56	78.33	51.67	88.54
2	voyage-lite-02-instruct		1024	4000	67.13	79.25	52.42	86.87
3	GritLM-7B	14.48	4096	32768	66.76	79.46	50.61	87.16
4	e5-mistral-7b-instruct	14.22	4096	32768	66.63	78.47	50.26	88.34
5	GritLM-8x7B	93.41	4096	32768	65.66	78.53	50.14	84.97
6	echo-mistral-7b-instruct-last	14.22	4096	32768	64.68	77.43	46.32	87.34
7	mxbai-embed-large-v1	0.67	1024	512	64.68	75.64	46.71	87.2
8	UAE-Large-V1	1.34	1024	512	64.64	75.58	46.73	87.25
9	text-embedding-3-large		3072	8191	64.59	75.45	49.01	85.72
10	voyage-lite-01-instruct		1024	4000	64.49	74.79	47.4	86.57

Models are a cost, datasets are an investment

- Working with unnecessary large models is a waste of time and money
- Right-sized models deliver faster ROI, making them easier to replace
- Invest in building high-quality datasets, and apply them to small models
- Your datasets will pay dividends for years, possibly decades

Don't fall in love with models: they're expendable. Fall in love with data!

LLMs from the trenches

- Retrieval Augmented Generation
  - Fresh domain knowledge comes from an external source of truth, not from the model
  - The model is mostly a writing assistant ➡️ small models work well (Phi-2 2.7B, TinyLlama 1.1B)
  - 100K context? Really? The typical novel is 50K-150K words.... Latency? Cost? 🤪 🤪 🤪
- Model customization
  - The cost of fine-tuning has dropped 1000x (LoRA, QLoRA)
  - Lots of interest in model merging, but still very much experimental
- PoC ➡️ Production
  - Cost/performance is everything: define latency, throughput, and ROI per project
  - Inference is 80-90% of the cost, not training
- Not just Nvidia GPUs
  - AMD MI300 and custom accelerators (AWS, Google, Intel Habana) have better cost/performance
  - Mid-range GPUs work nicely on small models (Nvidia A10G, aka g5 instances on AWS)
  - Local CPU inference local is quickly becoming a thing (Intel Meteor/Lunar Lake, AMD Ryzen AI)

Improving cost/performance with model acceleration

New Attention layers

MQA, GQA, SWA

Framework features

Compilation, quantization, model merging

Faster Attention layers

Flash Attention, Paged Attention

Hardware features

CPU, GPU, custom accelerators

Deep dives on youtube.com/juliensimonfr

Accelerating Hugging Face models with PyTorch 2

https://pytorch.org/blog/Accelerating-Hugging-Face-and-TIMM-models/ (12/2022)

One-line optimization for CPU and GPU

```
import torch
from transformers import BertTokenizer, BertModel

device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained("bert-base-uncased").to(device=device)
model = torch.compile(model, backend="inductor")
text = "Replace me by any text you'd like."
encoded_input = tokenizer(text, return_tensors='pt').to(device=device)
output = model(**encoded_input)
```

```
from diffusers import DiffusionPipeline
import torch

pipe = DiffusionPipeline.from_pretrained("runwayml/stable-diffusion-v1-5").to("cuda")
pipe.unet = torch.compile(pipe.unet)
images = pipe(prompt, num_inference_steps=steps, num_images_per_prompt=batch_size).images[0]
```

Accelerating Hugging Face models on AMD GPUs

https://github.com/huggingface/optimum-amd

```python
import torch
from transformers import AutoModelForCausalLM, AutoTokenizer, LlamaForCausalLM

tokenizer = AutoTokenizer.from_pretrained("tiiuae/falcon-7b")

with torch.device("cuda"):
    model = AutoModelForCausalLM.from_pretrained(
        "tiiuae/falcon-7b",
        torch_dtype=torch.float16,
        use_flash_attention_2=True,
)
```

Zero code change

AMD Instinct MI210/MI250/MI300, more coming

All Hugging Face models are supported

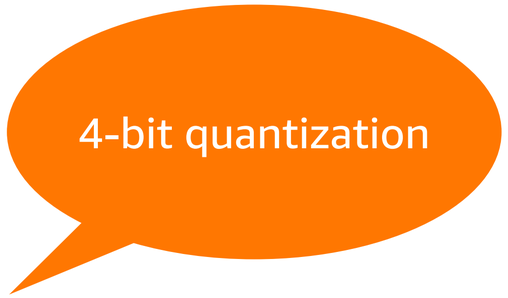
Also: ONNX, Flash Attention 2 (benchmarks), GPTQ, AWQ, TGI

Accelerating Hugging Face models on Intel CPUs

https://github.com/huggingface/optimum-intel + https://huggingface.co/docs/optimum
https://github.com/huggingface/optimum-intel/tree/main/examples

- Optimum Intel library: simple transformers-like API and CLI based on:
  - Intel Neural Compressor https://github.com/intel/neural-compressor
  - Intel OpenVINO https://github.com/openvinotoolkit/openvino
- Static and dynamic post-training quantization, quantization-aware training

```
from transformers import AutoTokenizer, pipeline
from optimum.intel import OVModelForCausalLM, OVWeightQuantizationConfig

model_id = "microsoft/phi-2"
device = "gpu" # Intel integrated GPU (iGPU)

# Create the quantization configuration with desired quantization parameters
q_config = OVWeightQuantizationConfig(bits=4, group_size=128, ratio=0.8)
# Create OpenVINO configuration with optimal settings for this model
ov_config = {"PERFORMANCE_HINT": "LATENCY", "CACHE_DIR": "model_cache", "INFERENCE_PRECISION_HINT": "f32"}

model = OVModelForCausalLM.from_pretrained(
    model_id,
    export=True,
    quantization_config=q_config,
    device=device,
    ov_config=ov_config,
  )
```



Phi-2 2.7B on an Intel-based laptop

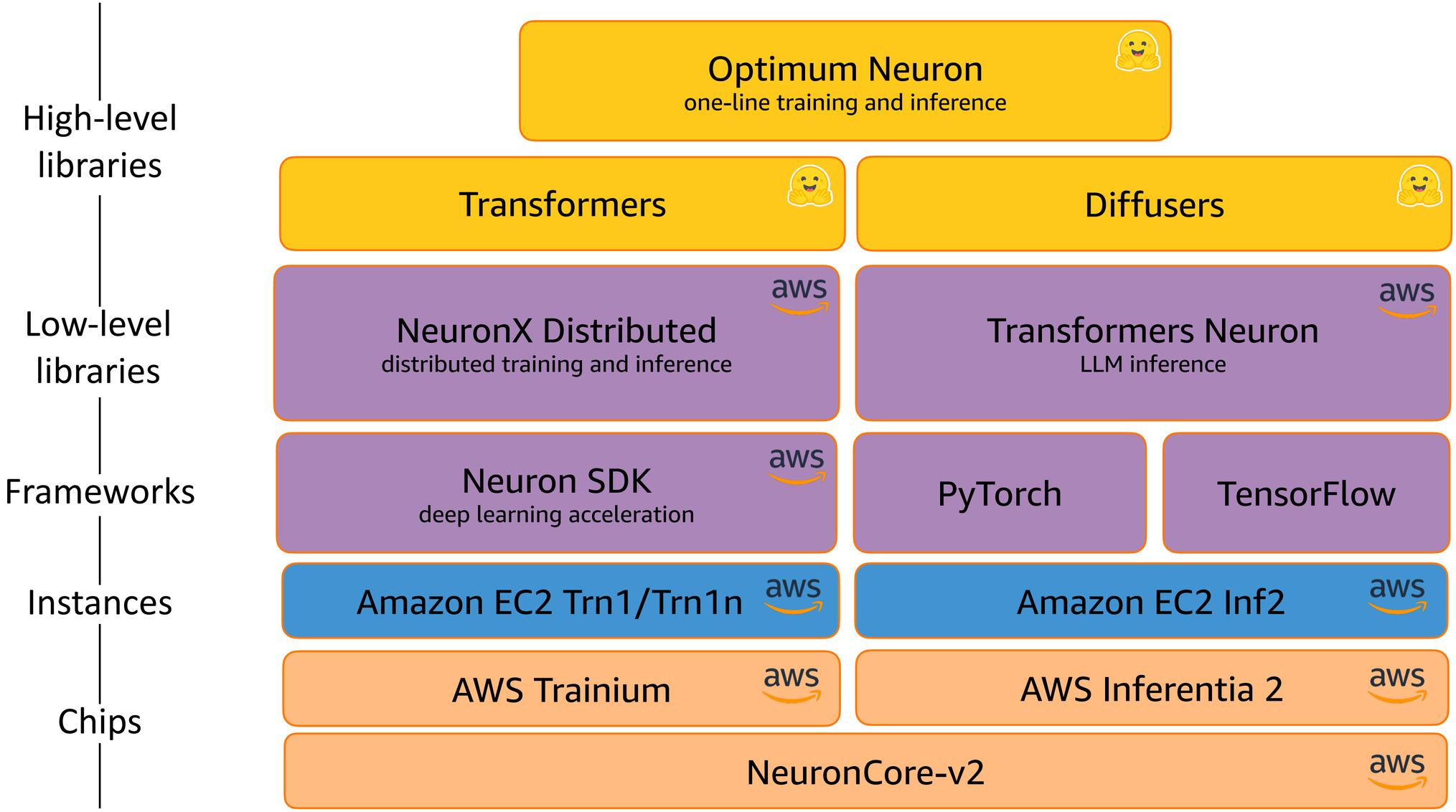
https://huggingface.co/blog/phi2-intel-meteor-lake

4-bit quantization with Optimum Intel

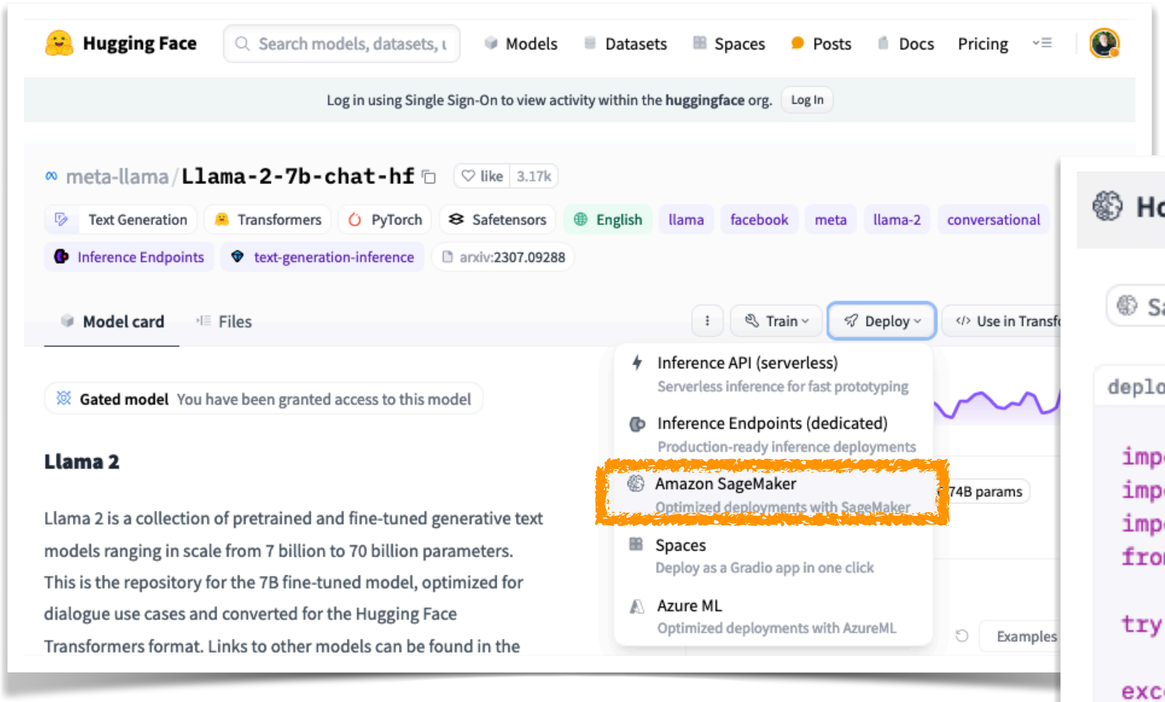
MSI Prestige 13 AI Evo
$1280 (amazon.com)

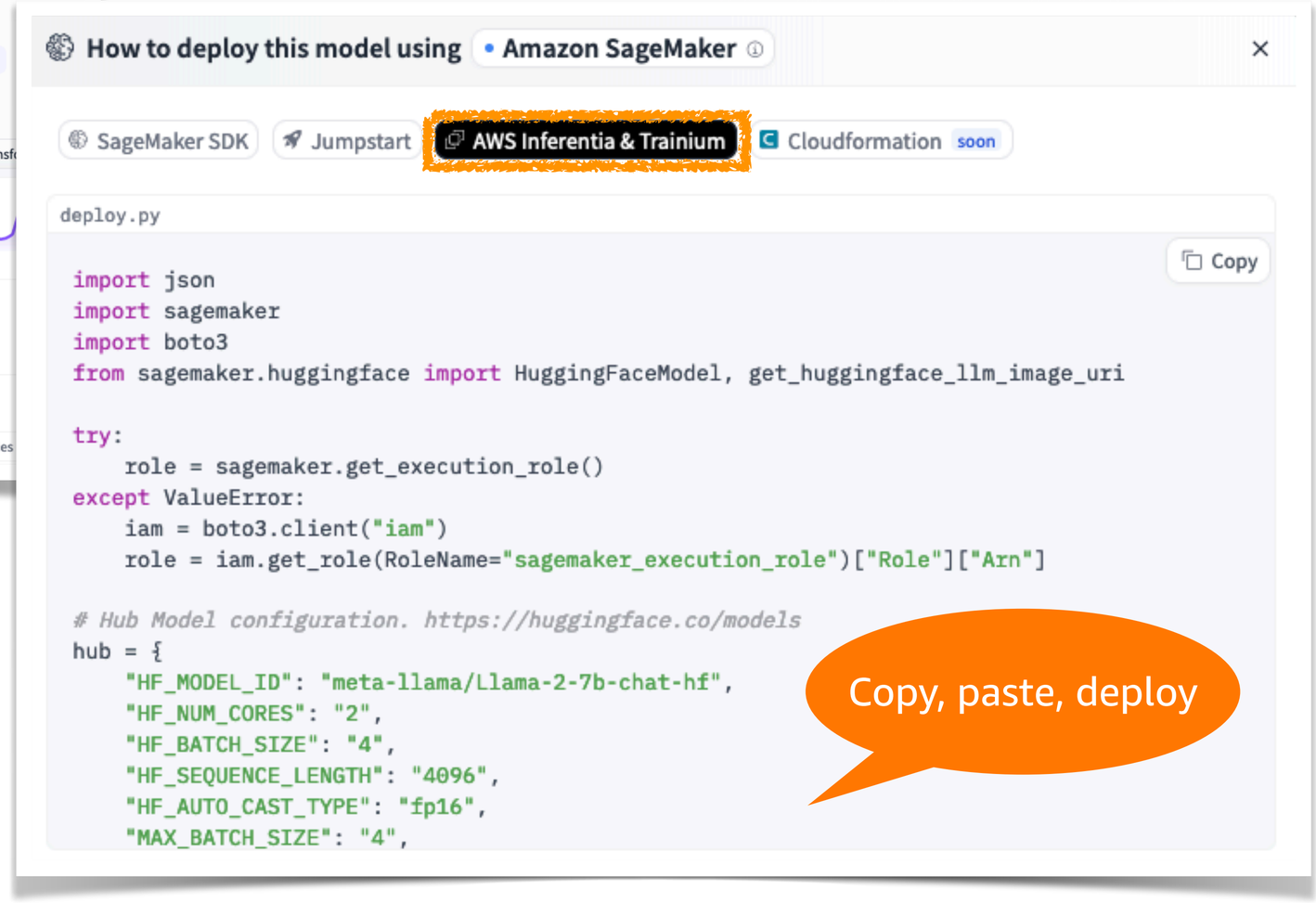
Hugging Face models on AWS accelerators

Not an architecture graph :)



Deploying LLMs on AWS Inferentia2





List of pre-compiled models:
https://huggingface.co/aws-neuron/optimum-neuron-cache/tree/main/inference-cache-config

```
$ optimum-cli neuron cache lookup <model_id>
```

Summing things up

- Hugging Face models are the de facto standard for AI apps.
- Don't believe the hype: today's "best" model will be superseded in weeks
- No model rules them all : find the most appropriate one for each use case
- Small open-source models combined with quality data are the way to go
- LLM acceleration helps you maximize performance and minimize cost

The future is open

Julien Simon, Chief Evangelist, Hugging Face
julsimon@huggingface.co
youtube.com/juliensimonfr