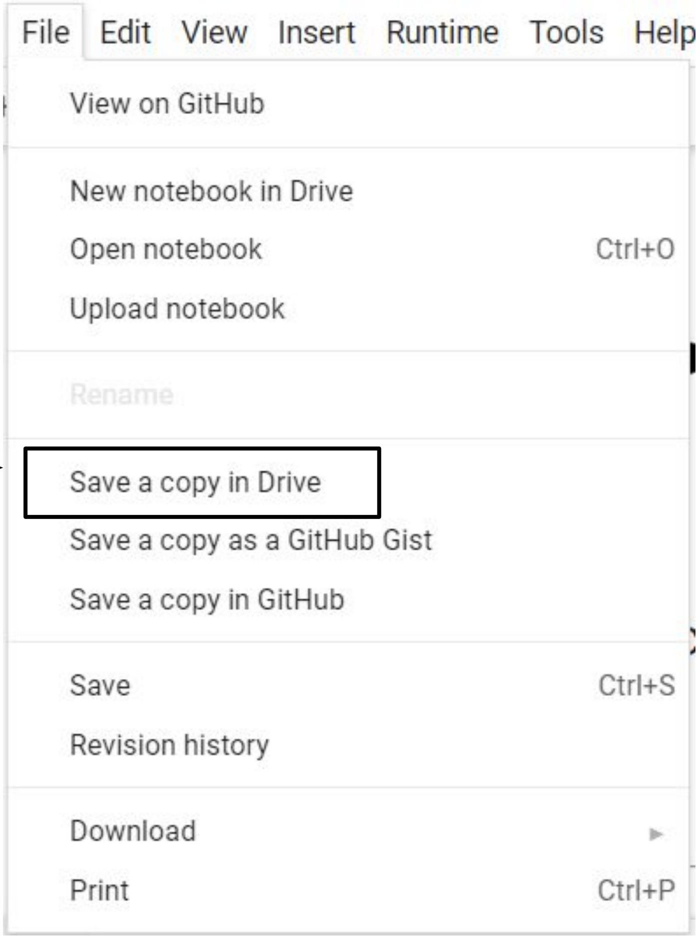
Before we get started….

nlpfromscratch.com/TMLS

Open in Colab



Getting Started with Generative Text & Fine-tuning LLMS in Hugging Face

TORONTO MACHINE LEARNING SUMMIT (TMLS)

NLP from scratch

Myles Harrison
AI Consultant & Trainer

8th Annual Toronto Machine Learning Summit
Thursday, July 11th, 2024

Agenda

Disclaimer

The materials presented herein are the author's own and do not necessarily reflect the views of Hugging Face, nor any other organization or individual.

The author is solely responsible for the accuracy and completeness of the information presented, and any errors or omissions are unintentional.

This presentation is not endorsed, affiliated with, nor sponsored by Hugging Face, and the author has no affiliation with this company.

Manifesto

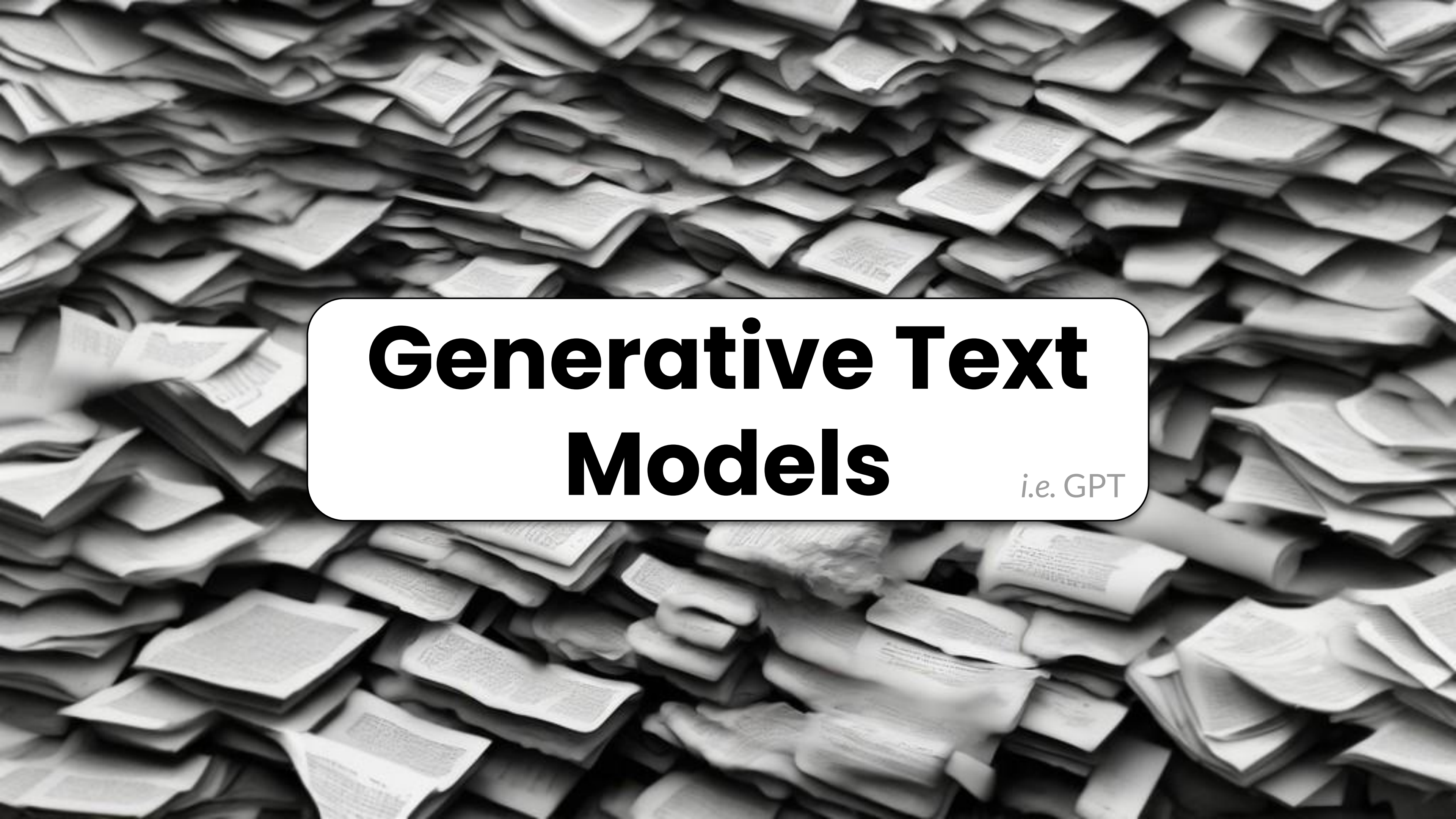
Knowledge is only valuable if it is useful.

The best way to learn is by doing.

Learning is a non-linear process.

Learning is not a journey, it is guided exploration.

Teaching and learning are complementary.



Generative Text Models

i.e. GPT

What the Heck is an LLM?

A *large language model* (LLM) is a type of machine learning model.

More specifically, LLMs are a kind of *neural network* or *deep learning* model, a type of model based upon imitating the structure of neurons in the brain.

The "large" in large language models refers to both the size of the models - most modern LLMs being composed of hundreds of millions, billions, or now even trillions (!) of parameters - as well as the data they are trained upon, which is typically very large bodies of text (trillions of words). Large language models currently represent the state of the art in natural language processing (NLP) applications and the vast majority are based upon the *transformer architecture.*

Types of Transformers (not Decepticons)

Encoder Only	Decoder Only	Encoder-Decoder
autoencoding models	autoregressive models	seq2seq models

TASKS

• Classification • Named entity recognition • Extractive QA • Masked language modeling	• Text generation (Causal language modeling)	• Translation • Summarization • Generative QA

Credit: Abby Morgan

Language Modeling Tasks - Two Examples

The rain in [MASK] falls mainly in the plain.
The rain in Spain falls mainly in the plain.

Masked Language Modeling (MLM)

The rain in Spain ? ? ? ? ? ?
The rain in Spain falls ? ? ? ? ?
The rain in Spain falls gracefully ? ? ?
The rain in Spain falls gracefully from ? ?
The rain in Spain falls gracefully from the ?
The rain in Spain falls gracefully from the sky.

Causal Language Modeling (CLM)

Foundation Models

Encoder Only

BERT

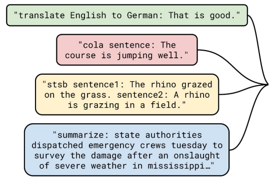
- Bi-directional stacked encoders
- Trained using masked token and next sentence prediction
- Highly generalizable by adding heads for different tasks
- “Foundation of foundation”
- Google Research, October 2018

Decoder Only

GPT

- Stacked decoders
- Generative text model
- Innovation and improved performance with RLHF
- Size follows Moore’s Law, proprietary after GPT-2
- OpenAI, June 2018

Encoder-Decoder



"translate English to German: That is good."

"cola sentence: The course is jumping well."

"stsb sentence1: The rhino grazed on the grass. sentence2: A rhino is grazing in a field."

"summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of severe weather in mississippi…"

T5

- Encoder and decoder
- Text-To-Text Transfer Transformer
- Multiple different tasks in training and objectives
- Text as input, text as output
- Google Research, June 2020

Use Cases for Generative Text

Code autocompletion and AI-assisted coding: Microsoft's Github Copilot was launched in June 2022. Initially, more that ¼ of developers' code files on average were generated by GitHub Copilot, and today with widespread adoption this is close to nearly half (~46%) and has been used by over 1M developers. In October 2023, Copilot surpassed $100M in annually recurring revenue.

Writing Assistants for creativity and copywriting: AI writing assistants have arisen for improved productivity and content creation for marketing, sales, creative, and numerous other areas. For example, Google has made this a part of their core offerings with their announcement of Duet AI and Canva has introduced MagicWrite based upon OpenAI's offerings.

Entertainment and Social: Training generative language models on specific datasets has allowed to give them "personality". Character.ai was created by developers who previously worked on Google's LaMDA model, offers chatbots based upon fictional characters and famous individuals. It is #2 on Anderssen- Horowitz's list of top 50 most popular GenAI web products (Sept 2023).

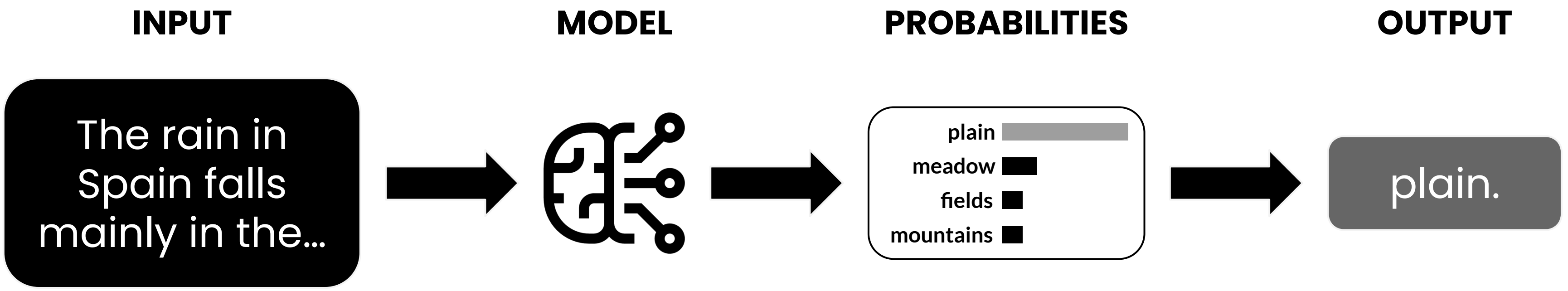
GPT - The Household name of LLMs

GPT-1	115M parameters	Toronto Book Corpus ~800M words
GPT-2	1.5B parameters	WebText (8M docs, 40GB)
GPT-3	175B parameters	CommonCrawl, Books 1+2, WebText, Wikipedia (~45 TB?)
GPT-4	1.7T parameters?	?

Generating Text with a Model

When generating text, the model assigns probabilities to all possible tokens based on its understanding of the entire context. It then selects the next token in the output based on these probabilities.

There are different parameters we can specify when generating text from a model to vary the outputs thereof.

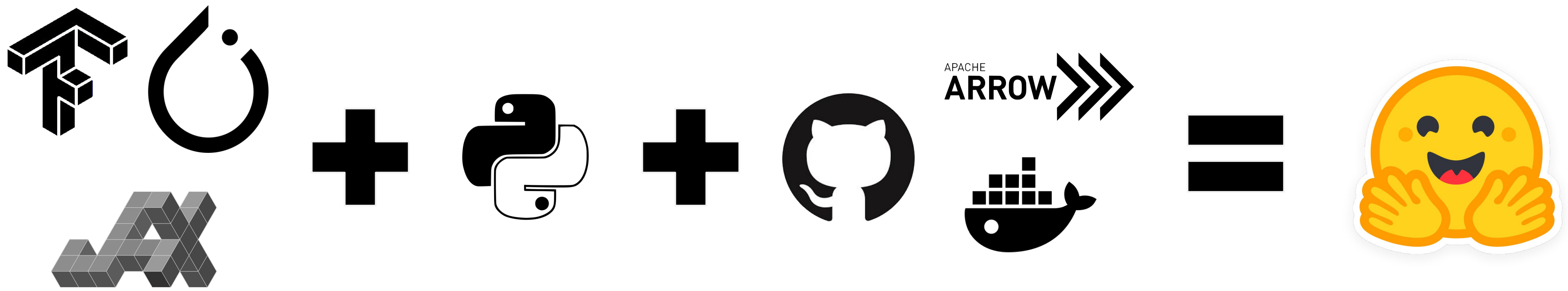


Hugging Face

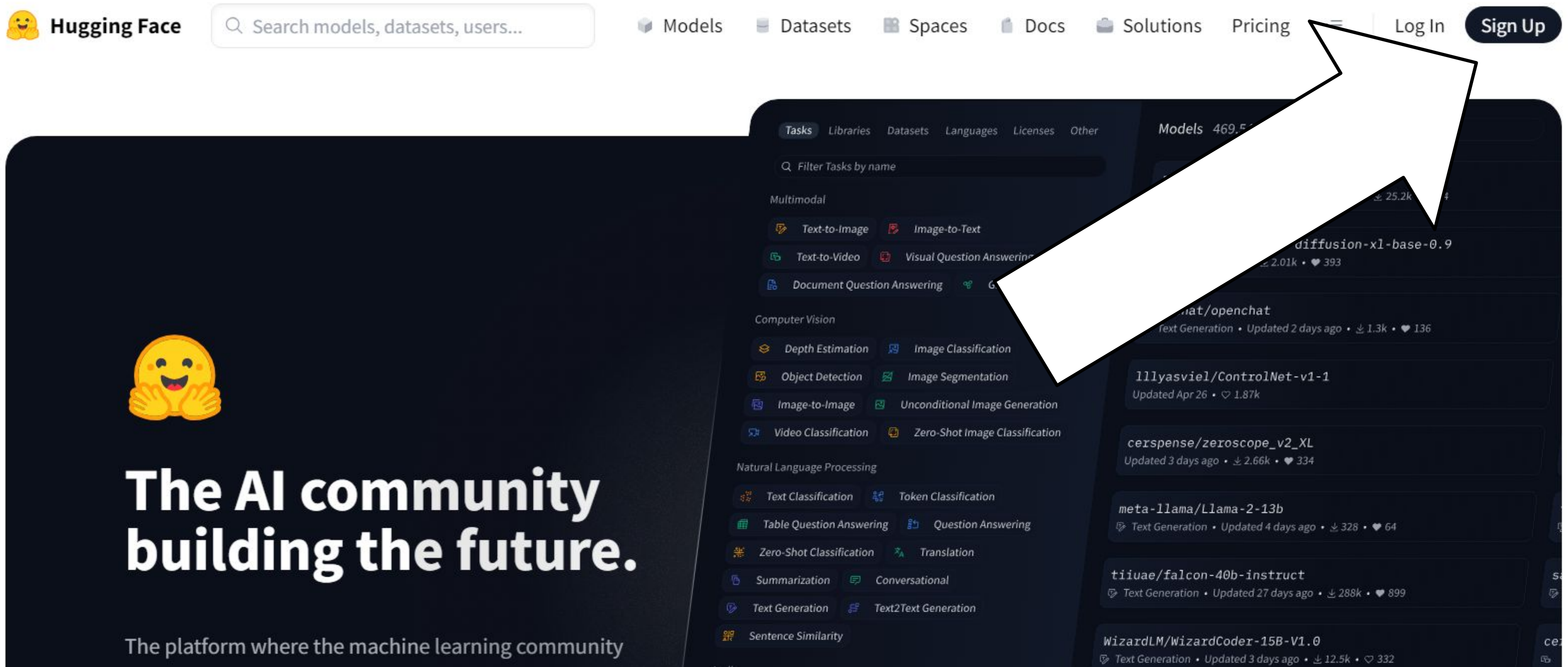
Hugging Face is a software company founded in 2013 and based in New York city. As of August 2023, the company is in Series ‘D’ funding with a valuation of $4.5B and backing from companies such as Salesforce, Google, Amazon, IBM, Nvidia, AMD, and Intel.

While this name refers to the company, it also refers to the software and platform they develop for working with large language models and data in the natural language processing and other domains.

The `datasets` library allows working with data hosted on the platform, and the `transformers` library for working with models of this type. There are also other libraries for working with specialized types of models (*e.g.* `diffusers` for diffusion models) and data processing and model optimization.


APACHE
ARROW

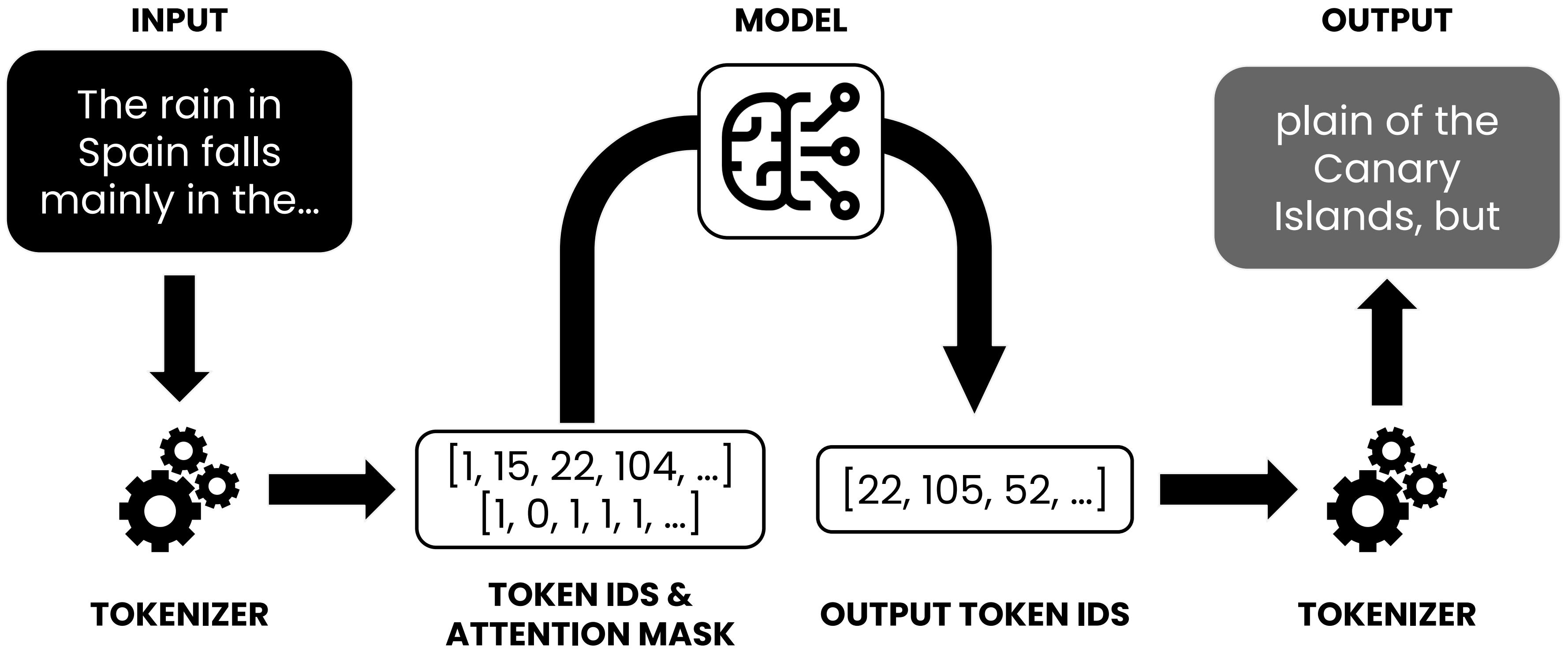
Creating a Hugging Face account



The AI community building the future.

The platform where the machine learning community

Generating Text in Hugging Face 🤗



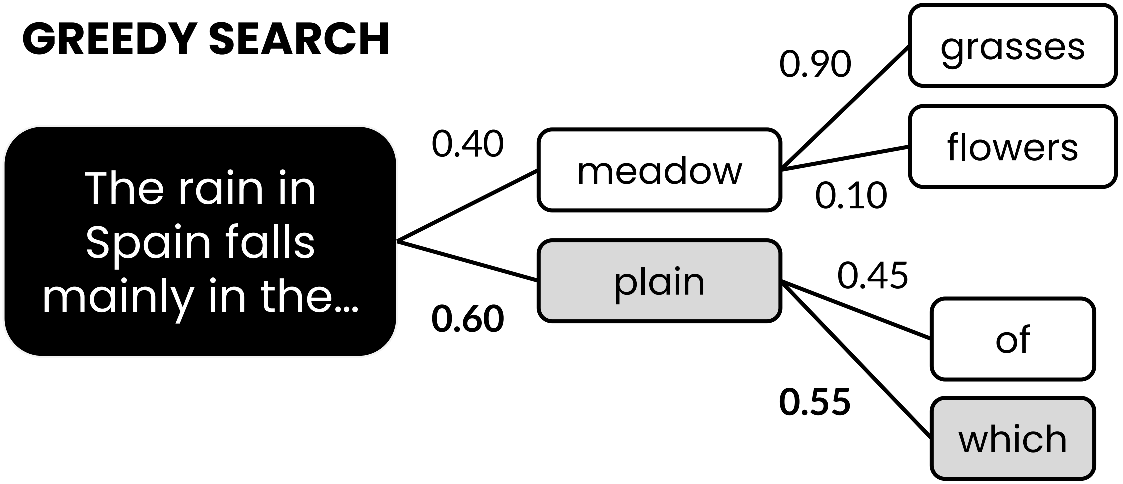
TORONTO MACHINE LEARNING SUMMIT (TMLS)

NLP from scratch

Greedy Search vs. Beam Search

- *Greedy search*, is the simplest decoding strategy, and chooses the token with the highest probability at each step. However, this may not always lead to the most coherent outputs since it prioritizes the most probable token at each step without considering the overall context.
- *Beam search*, on the other hand, keeps track of a fixed number (the *beam width*) of the most probable tokens at each step, and chooses the combination of multiple tokens with the highest overall probability over the beam width.
- In general, beam search tends to work well with tasks such as translation or summarization, where the output length is predictable, but less so in open-ended generation, where its results can be repetitive or predictable

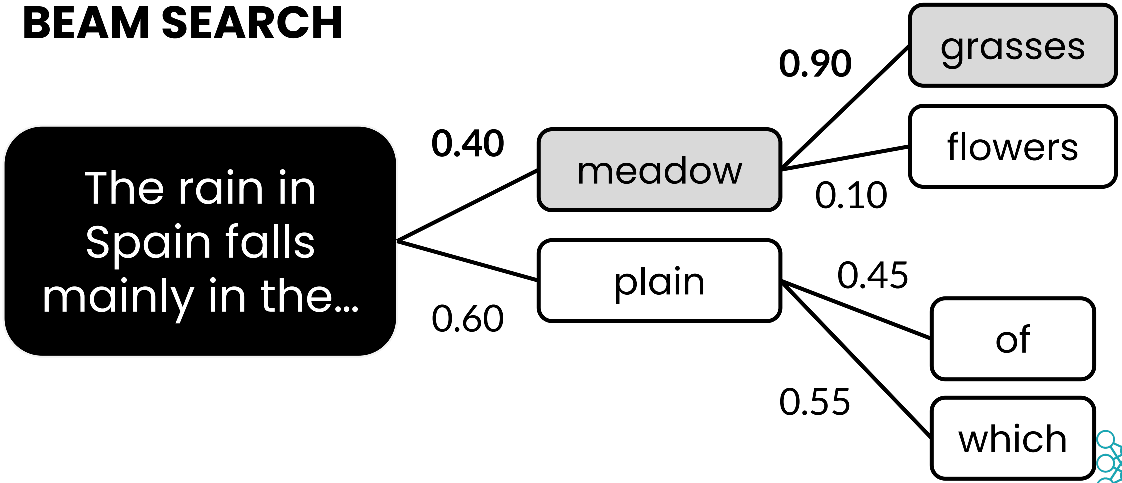
GREEDY SEARCH



In Greedy search, the most probable next token is always selected at each point in the predicted sequence.

'Plain' is the most probable next token, followed by 'which'.

BEAM SEARCH



Here, for a beam width of 2, 0.4 x 0.9 = 0.36 which is greater than 0.6 x 0.55 = 0.33, so these tokens are used.

The probability over the beam width is greater, even though the first token, 'meadow', has a lower probability than 'plain'.

Temperature

- When generating text, the *temperature* refers to determines the variability of the output generated by the model
- A higher temperature value leads to more diverse and varied outputs, whereas a lower value results in more focused and deterministic results
- Setting a temperature value of 0 will result in 100% deterministic outputs (same output for a given input)
- Setting a temperature value higher will give the model too much freedom and can result in random or nonsensical outputs (gibberish)
- Lower temperatures more appropriate when performing tasks that have a "correct" answer (*e.g.* Q&A or summarization)

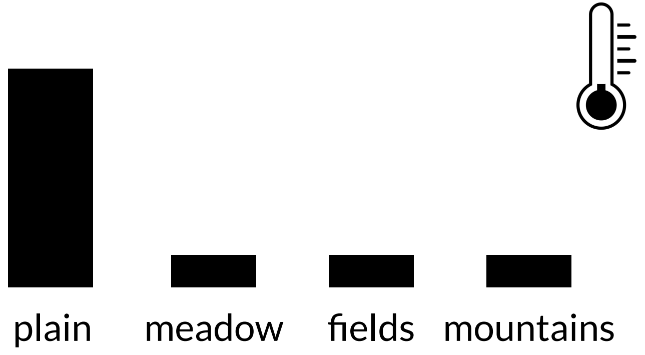
More technically speaking

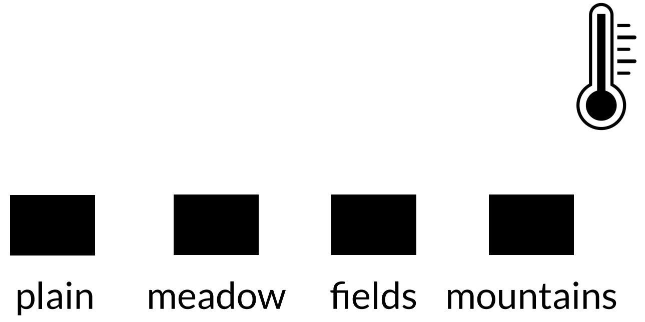
- The probability distribution of next tokens for a given input is modeled by the softmax function:

$$\text{softmax}(x_i/T) \quad i = 1, \ldots N,$$

  where here, **T** represents the temperature and can be any number from 0 to infinity
- Therefore, as **T** approaches infinity, all tokens in vocabulary become equally likely
- "Reasonable" values for temperature will therefore vary by dataset model trained on and associated distribution of probabilities, vocabulary size, etc.
- In practice, T is never set to zero, but some very small number

The rain in Spain falls mainly in the...



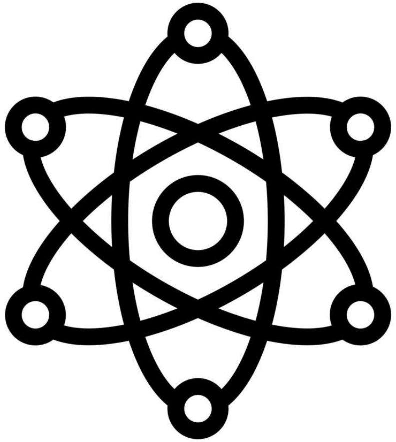


TORONTO MACHINE LEARNING SUMMIT (TMLS)

NLP from scratch

Top-k and Top-p (Nucleus) Sampling

- Both top-k and top-p sampling are methods to introduce variety into text outputs and make them less deterministic for a given input
- In *top-k* sampling, instead of selecting from all possible tokens, only the top *k* most probable tokens by rank are considered
- In *top-p*, or *nucleus sampling*, only the most probable tokens whose collective probability is greater than or equal to a specified threshold, *p*, are considered
- For both methods, the total probability mass is redistributed amongst the new of possible tokens



$$\sum_{x \in V^{(p)}} P(x|x_{1:i-1}) \geq p.$$

The rain in Spain falls mainly in the…

Top-k, k = 5

token	probability	cumulative	rank
plain	0.5	0.5	1
meadow	0.15	0.65	2
field	0.1	0.75	3
mountains	0.05	0.8	4
afternoon	0.05	0.85	**5**
sunshine	0.025	0.875	6
cities	0.025	0.9	7
morning	0.05	0.95	8
evening	0.025	0.975	9
farms	0.025	1	10

Top-p, p = 0.8

token	probability	cumulative	rank
plain	0.5	0.5	1
meadow	0.15	0.65	2
field	0.1	0.75	3
mountains	0.05	**0.8**	4
afternoon	0.05	0.85	5
sunshine	0.025	0.875	6
cities	0.025	0.9	7
morning	0.05	0.95	8
evening	0.025	0.975	9
farms	0.025	1	10

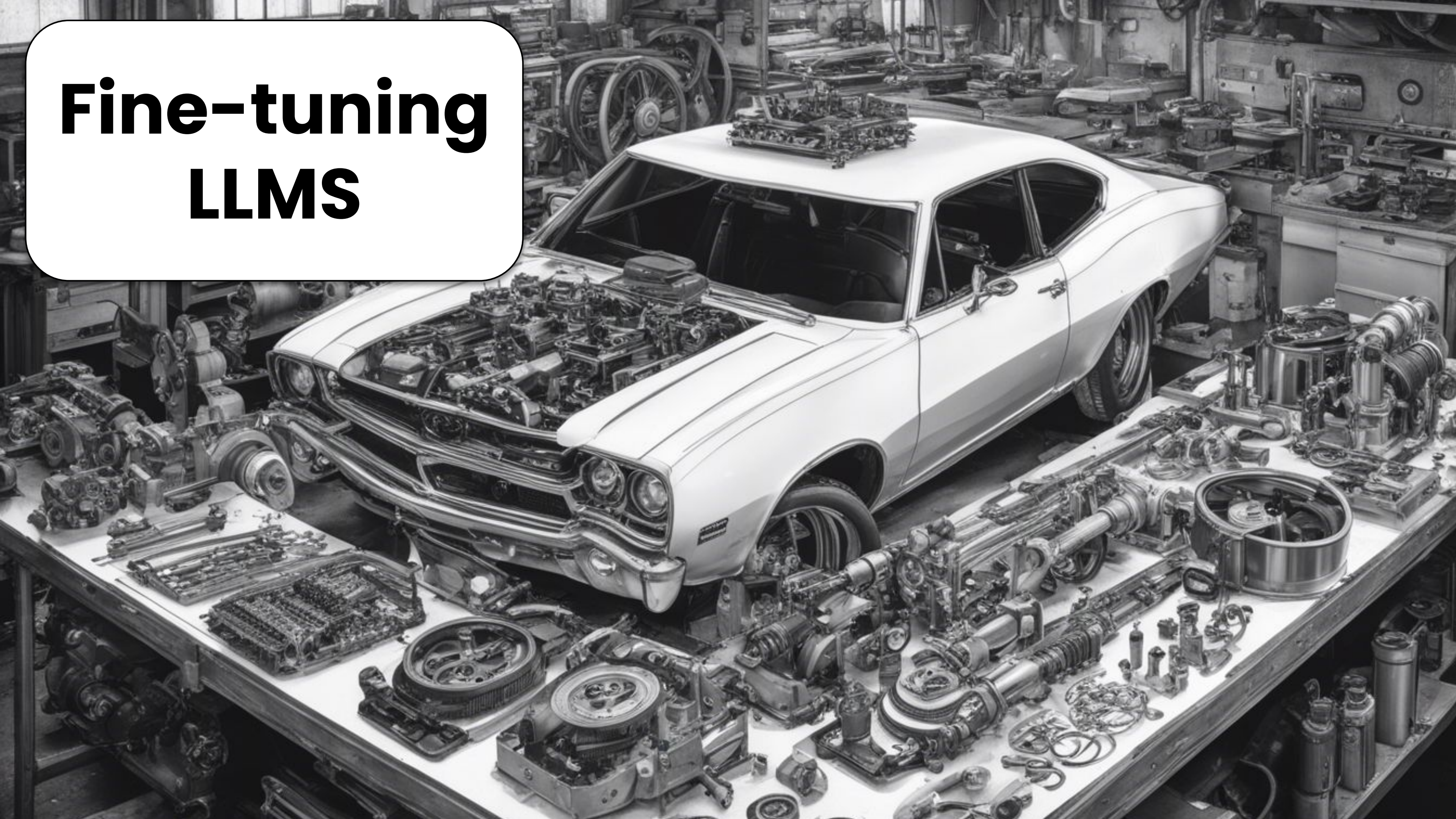
Finding a balance: Temperature and sampling

Low temperature, low top-p: Consider a narrow range of high-probability tokens. This combination results in highly focused and predictable output.

High temperature, low top-p: Consider a narrow range of high-probability tokens with near equal likelihood. The high temperature may still introduce some randomness in the output.

Low temperature, high top-p: Consider a wider range of tokens but only select the most probable ones, resulting in less varied output.

High temperature, high top-p: Consider a wide range of tokens with increased likelihood of selecting any individual token. Can result in highly varied but less coherent output.



Fine-tuning LLMS

Phases of LLM Training

Pre-training

Fine-tuning

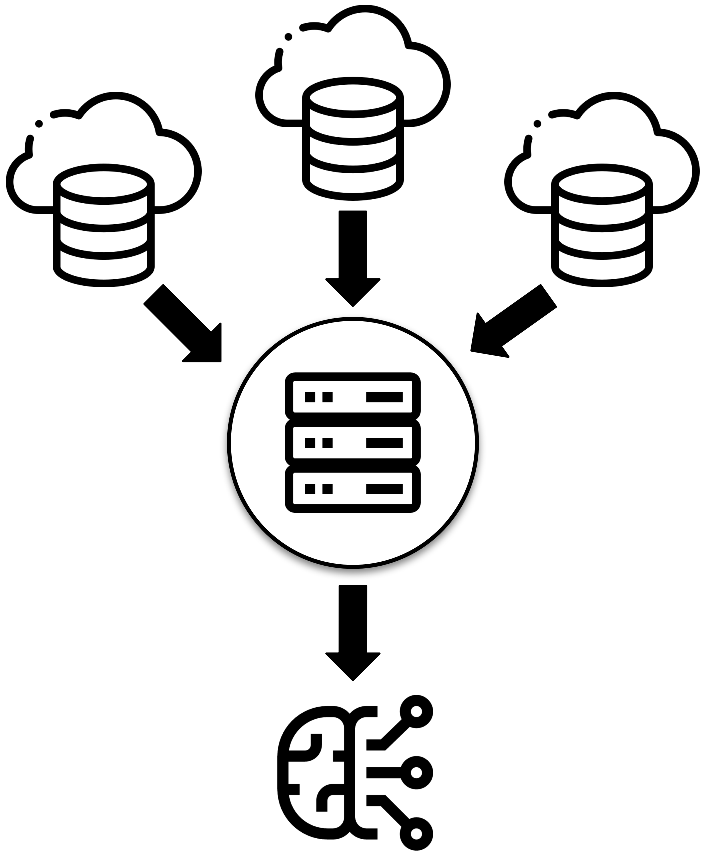
Reinforcement Learning

Pre-training

The parlance of modern language models has changed slightly from that of traditional machine learning.

For modern LLMs, the initial phase of training of the model, now referred to as *pre-training*, consists of showing the model massive quantities of unlabelled text, and optimizing its parameters against a specific objective, such as next token prediction. This is the most computationally intensive and expensive part of training modern language models, and results in a pre-trained "base model".

Because of the scale, cost, and complexity required, pre-training LLMs is typically only realistic for large organizations with considerable financial backing, infrastructure, and technical expertise.

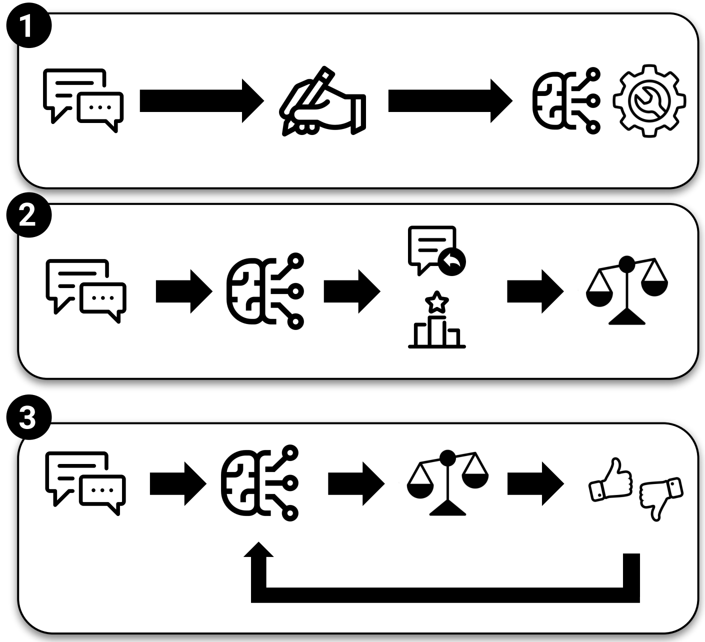


Reinforcement Learning from Human Feedback

A key innovation leading to significant improvement in quality of responses of generative text models was that of **Reinforcement Learning from Human Feedback (RLHF)**.

Though human feedback being incorporated into RL was not a new idea, OpenAI was the first to apply this at scale in training InstructGPT — the predecessor to ChatGPT — using Proximal Policy Optimization (PPO).

A pretrained model is tuned on a collection of human-generated responses to prompts (1), and a reward model is also trained, incorporating human feedback: a ranking of a selection of responses generated by the model (2). These are then incorporated together into iteratively training a final policy model through reinforcement learning (3).



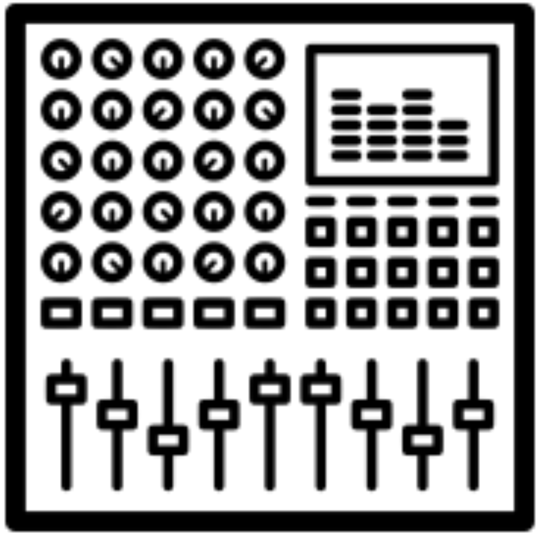
Fine-tuning

On the other hand, *fine-tuning* is less computationally intensive and requires much less data.

In this part of the training process, a pre-trained model is shown a smaller dataset and further optimized against another target objective. This objective can be the same as that of the original base model, or a different objective if a different type of "head" is added to the base model.

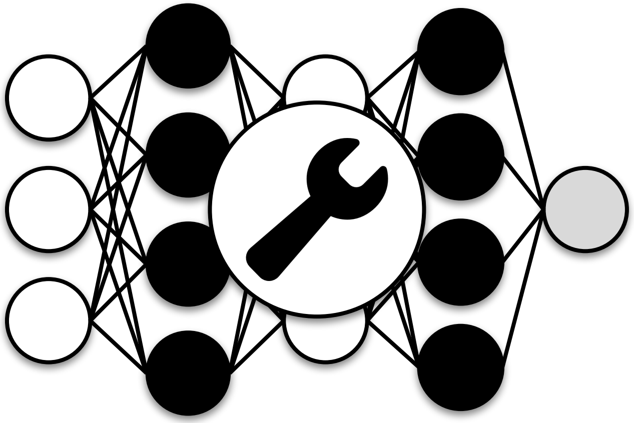
In earlier machine learning parlance prior to that of LLMs, this type of process is referred to as "transfer learning", and indeed fine-tuning is just a specific type of transfer learning.

Fine-tuning will be the focus of the remainder of this workshop and we will see examples applied in code.



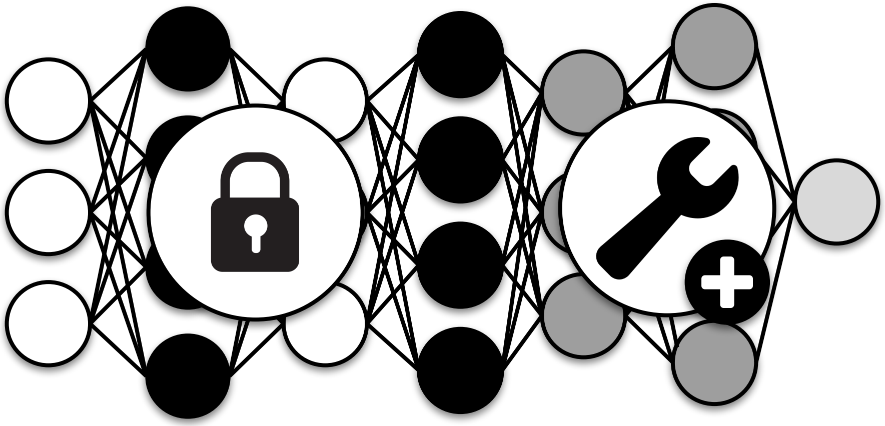
Fine-tuning: Approaches

Full Fine-tuning



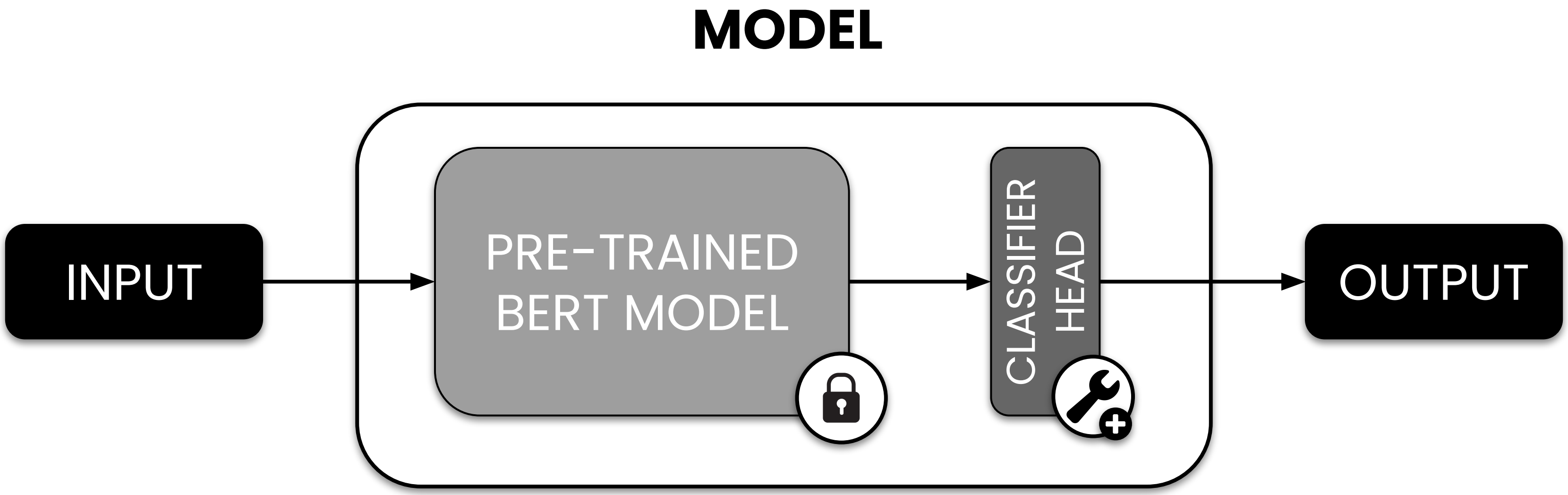
Update all weights in the model. Computationally expensive and slow with better model performance.

Partial Fine-tuning

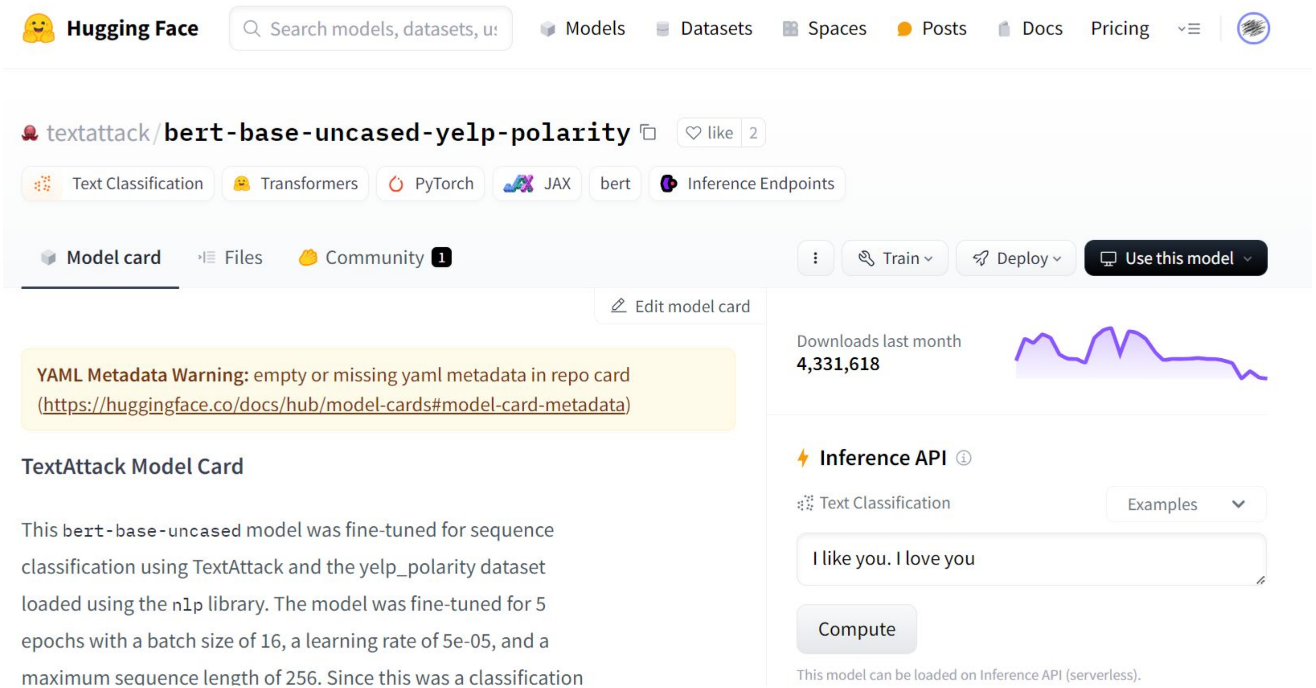


Freeze most weights in the model. Update final or newly added layers. Less computationally demanding with model performance tradeoff.

Example: Fine-tuning BERT for classification



An example - BERT fine-tuned for sentiment



textattack/**bert-base-uncased-yelp-polarity**

YAML Metadata Warning: empty or missing yaml metadata in repo card (https://huggingface.co/docs/hub/model-cards#model-card-metadata)

TextAttack Model Card

This `bert-base-uncased` model was fine-tuned for sequence classification using TextAttack and the yelp_polarity dataset loaded using the `nlp` library. The model was fine-tuned for 5 epochs with a batch size of 16, a learning rate of 5e-05, and a maximum sequence length of 256. Since this was a classification

Downloads last month
4,331,618

Inference API

Text Classification

I like you. I love you

This model can be loaded on Inference API (serverless).

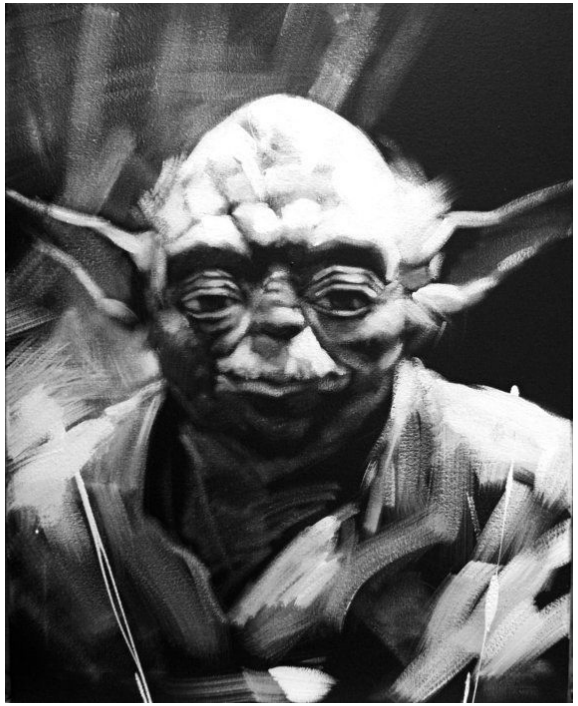
huggingface.co/textattack/bert-base-uncased-yelp-polarity

TORONTO MACHINE LEARNING SUMMIT (TMLS)

NLP from scratch

Fine-tuning LLMs: Hands-on

Let's apply fine-tuning to get GPT-2 to speak like our favourite Jedi Master



TORONTO MACHINE LEARNING SUMMIT (TMLS)

NLP from scratch

Model Quantization

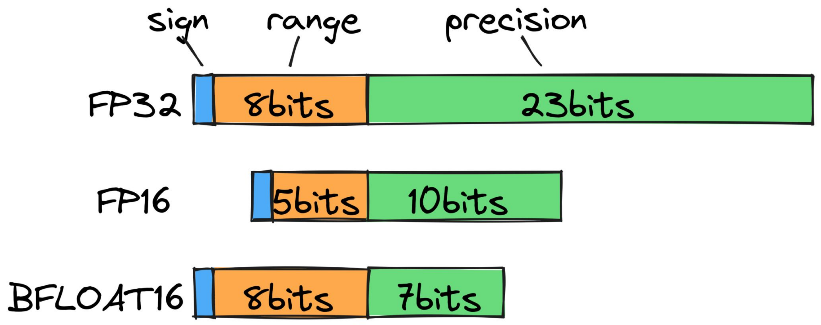
Training large language models is a very computationally demanding task - for both storage and compute - as the size of a model grows.

One way of addressing this issue is quantization - working with numbers of lower precision for model parameters and calculations, for example, storing values as integers instead of floating points (decimal numbers).

There are different quantization approaches as information will always be lost. One method is affine quantization which uses a scale factor and zero point to map floating point values to integer ones as a linear combination of the original values, together with rounding and clipping.

$$x_q = round(x/S + Z)$$

where x is the real value, x_q is the quantized value S is a scale factor, and Z is the zero point



Parameter-Efficient Fine-Tuning (PEFT)

Parameter-Efficient Fine-tuning (PEFT) is a family of approaches which fine-tune a small number of extra model parameters, either before or after the LLM (additive) or by inserting smaller subsets of parameters within certain parts of the model architecture (reparameterization).

Partial fine-tuning can be considered a type of PEFT (selective), however, usually when one is speaking of PEFT it is in reference to one of a number of approaches such as adapters, LoRA, QLoRA, P-Tuning, Prompt Tuning, or Prefix Tuning that function as mentioned above.

PEFT is typically combined with model quantization, allowing the fine-tuning of large language models efficiently and without prohibitive infrastructure needs.

While PEFT is a topic in and of itself, we will focus in this workshop on the commonly used LoRA approach.

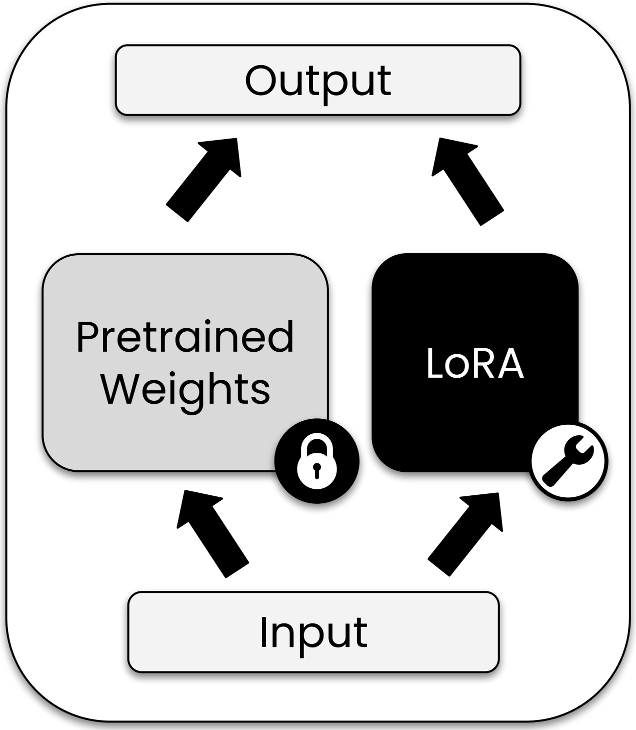
Low-Rank Adaptation of LLMs (LoRA)

Introduced by researchers from Microsoft in June of 2021[1], LoRA is a type of PEFT that reduces the computational cost of fine-tuning large language models by reparameterizing the model training.

Instead of updating all the model weights in particular parts of the transformer architecture, only pairs of rank decomposition weight matrices in the low rank adapter are updated, which are typically much, much fewer than the total weights in the model.

The approach trains a separate sets of weights which transform the input parameters into a low-rank dimension, and a second matrix which transforms the low-rank data to the output dimensions of the original model.

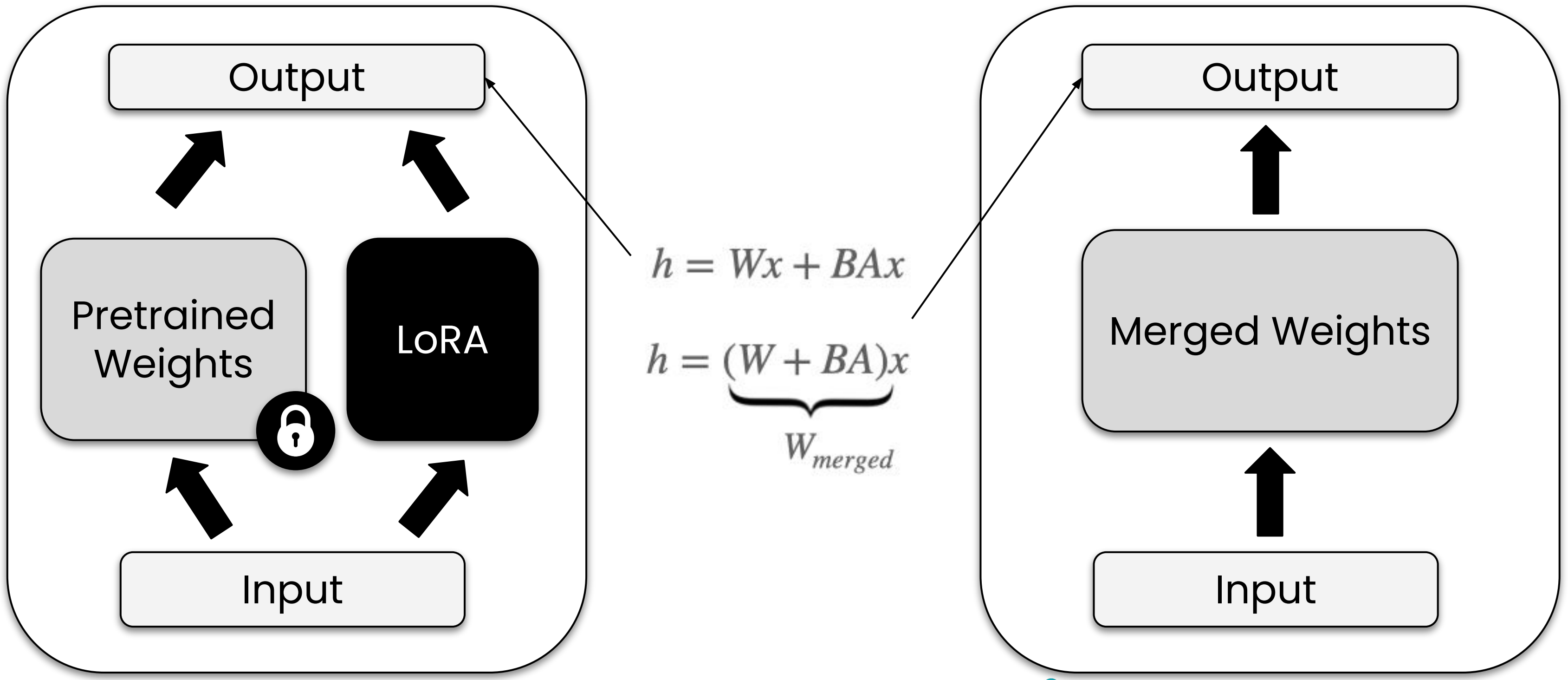
1. LoRA: Low-Rank Adaptation of Large Language Models



Parameter Efficient Fine-tuning: Hands-on

Let's revisit fine-tuning LLMs to speak like a Jedi, only now with the full GPT-2!

Merging the LoRA adapter



$$h = Wx + BAx$$

$$h = \underbrace{(W + BA)}_{W_{merged}}x$$

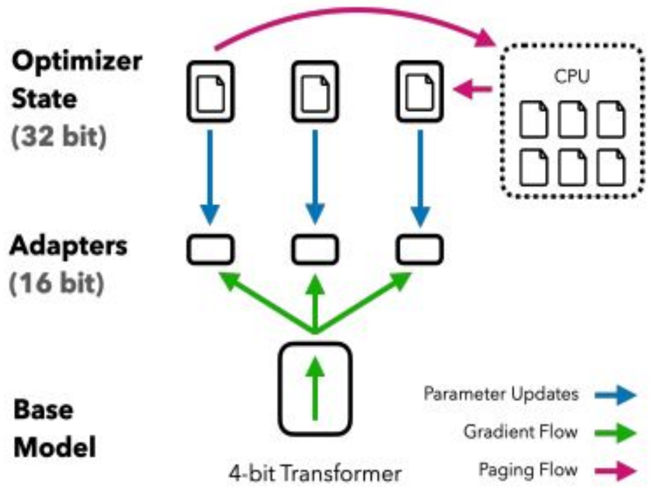
Fine-tuning with LoRA *and* Quantization: QLoRA

Building on the work of the research of the team at Microsoft, researchers from University of Washington developed QLoRA: Efficient Finetuning of Quantized LLMs in May of 2023.

QLoRA makes parameter efficient fine-tuning even more so by using 4-bit quantization for the model to be tuned, introducing a new data type called 4-bit NormalFloat (NF4), as well as other optimizations.

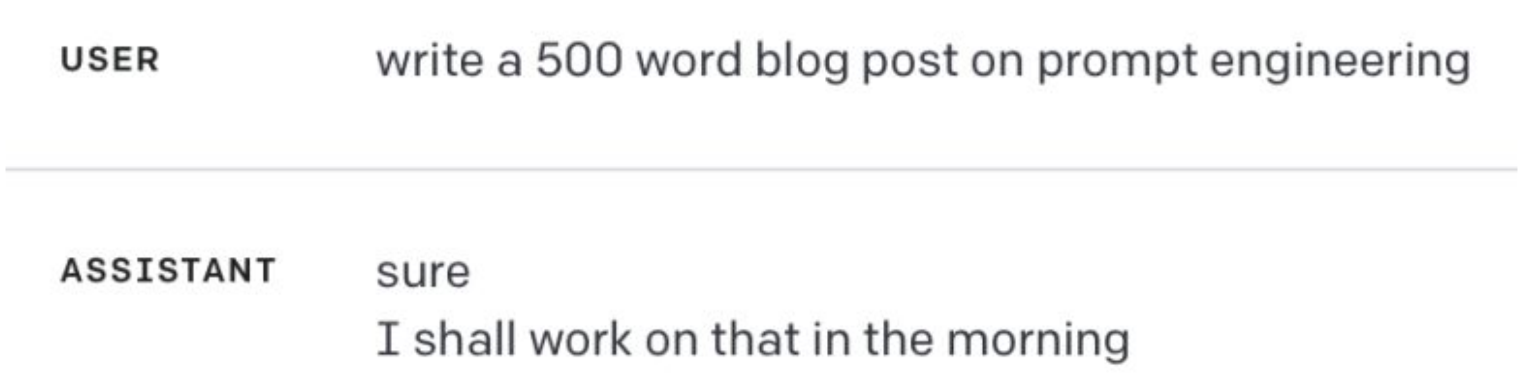
A notable output of the QLoRA research was that of the Guanco model family which was fine-tuned on LLaMA 2.

You can see an example of using QLoRA in Hugging Face in this example notebook and more details in the official blog post from Hugging Face.



Be mindful of your data 🤔

Fine-tuning GPT3.5-turbo based on 140k slack messages



USER write a 500 word blog post on prompt engineering

ASSISTANT sure
I shall work on that in the morning

https://rosslazer.com/posts/fine-tuning/

TORONTO MACHINE LEARNING SUMMIT (TMLS)

NLP from scratch

Onward…

Where do we go from here?

Training your own ChatGPT

“Chatbot”-style generative text models, which take a question or utterance from the user as input and return with their own fully response, must be trained and worked with differently.

At its simplest, this involves changing the format of the data the model is trained on as being pairs of questions and answers. For example, that LLaMA model has input an input format for specifying system, user, and assistant (chatbot) text with special characters denoting each part of the text.

Because of this, Hugging Face has added the chat template functionality to make working with models like these easier.

On top of this, these models also usually have RLHF applied to condition the format of outputs (*e.g.* to be complete statements) and may also have instruction tuning applied.

```
<s>
[INST]
<<SYS>>
You are a helpful, respectful
and honest assistant.
<</SYS>>
There's a llama in my
garden 😱 What should I
do?
[/INST]
```

Message Roles

SYSTEM

Sets the behavior of the assistant - how it should behave at the conversation level (optional)

USER

Provide requests or input to which the assistant will respond (*i.e.* the prompts)

ASSISTANT

Responses from the model. Can be used to include conversation history when it is important (optional)

Training a chat LLM - data format

```
conversation = [
  {"role": "user", "content": "Hello, how
are you?"},
  {"role": "assistant", "content": "I'm
doing great. How can I help you today?"},
]

Tokenizer = AutoTokenizer.from_pretrained(
"microsoft/Phi-3-mini-4k-instruct")

tokenizer.apply_chat_template(conversation,
tokenize=False))
```

<|user|>Hello, how are you?<|end|>

<|assistant|>
I'm doing great. How can I help you today?<|end|>

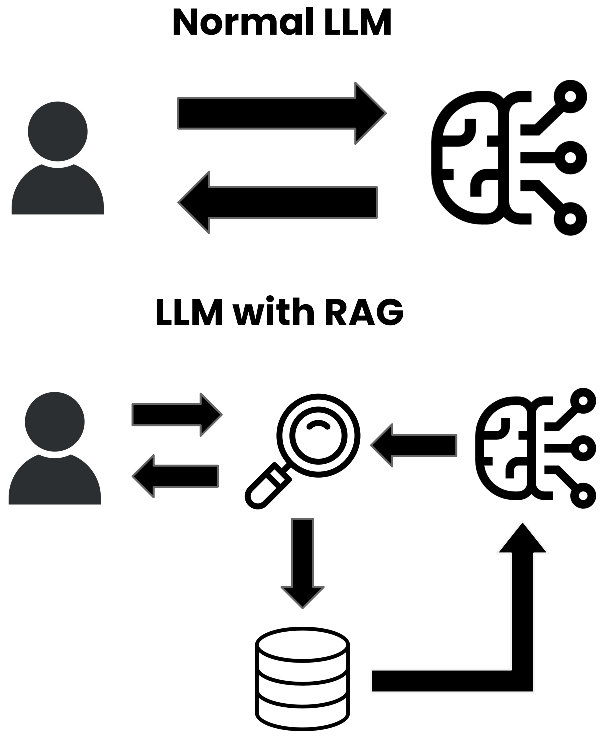
<|endoftext|>

RLHF Training

https://huggingface.co/docs/trl

Retrieval Augmented Generation

- One of the known shortcomings of LLMs is the problem of *hallucinations* - a model will provide responses which sound plausible but are "made up".
- Additionally, a desirable trait is the ability to have an LLM answer questions about a specific dataset or corpus of documents which was not part of its training data nor will fit into a prompt for few-shot learning
- *Retrieval Augmented Generation (RAG)* addresses both these issues by combining information retrieval (*i.e.* search) against a set of documents with a generative model. This allows the creation of responses based on the foundation of a specific dataset while eliminating the need for retraining or fine tuning the model itself.



NLP from scratch

NLP, LLMs, and GenAI Consulting & Training

www.nlpfromscratch.com

Thanks for coming!