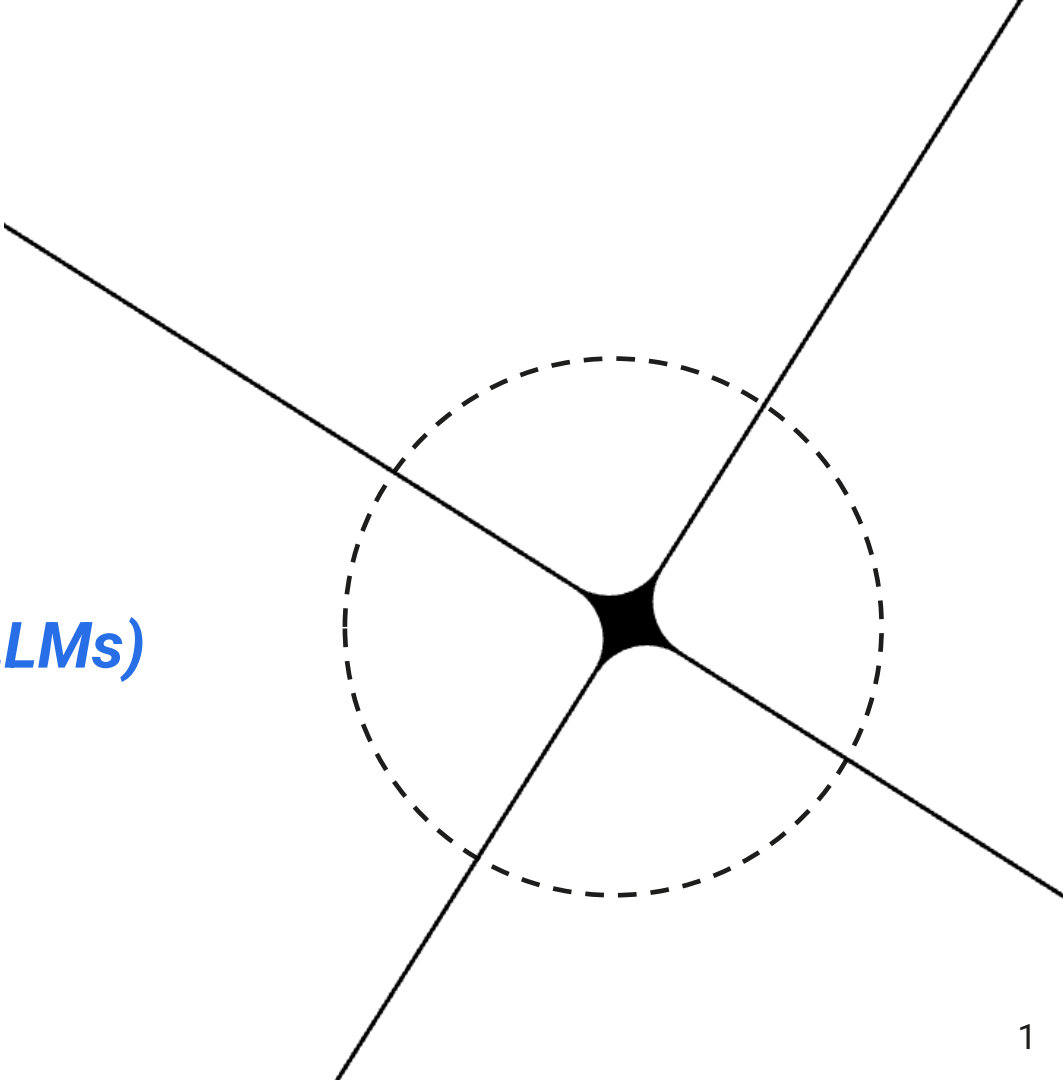
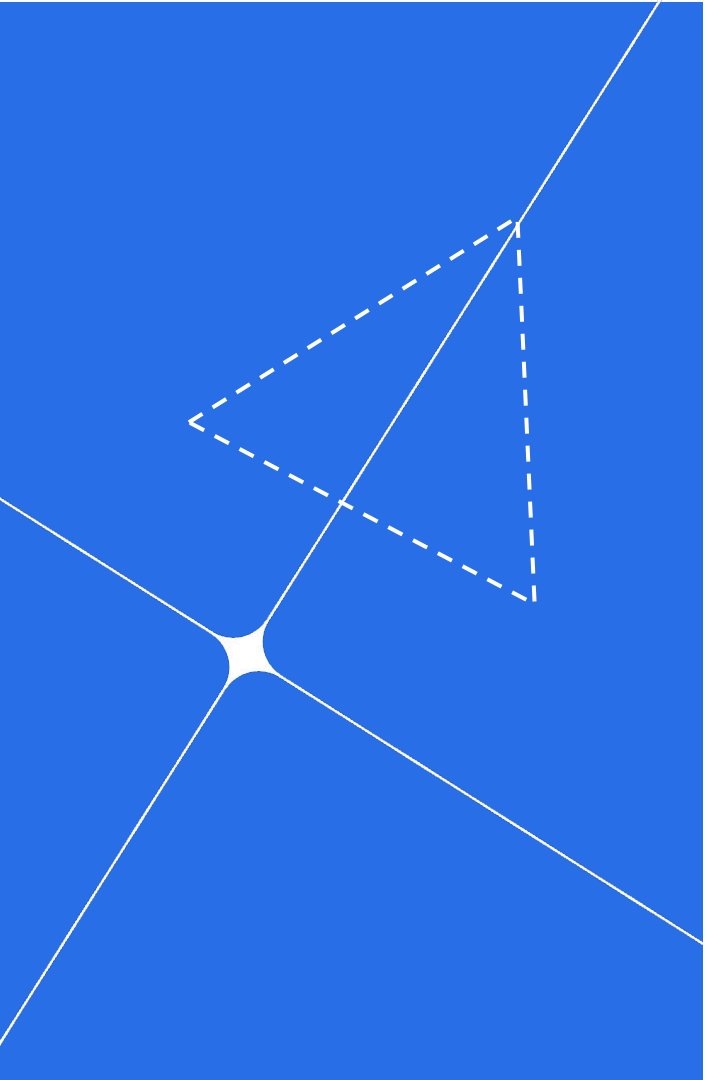
INSTITUT POLYTECHNIQUE DE PARIS

TELECOM SudParis

Training of Large Language Models (LLMs)

Luca Benedetto
(based on slides by Julien Romero)

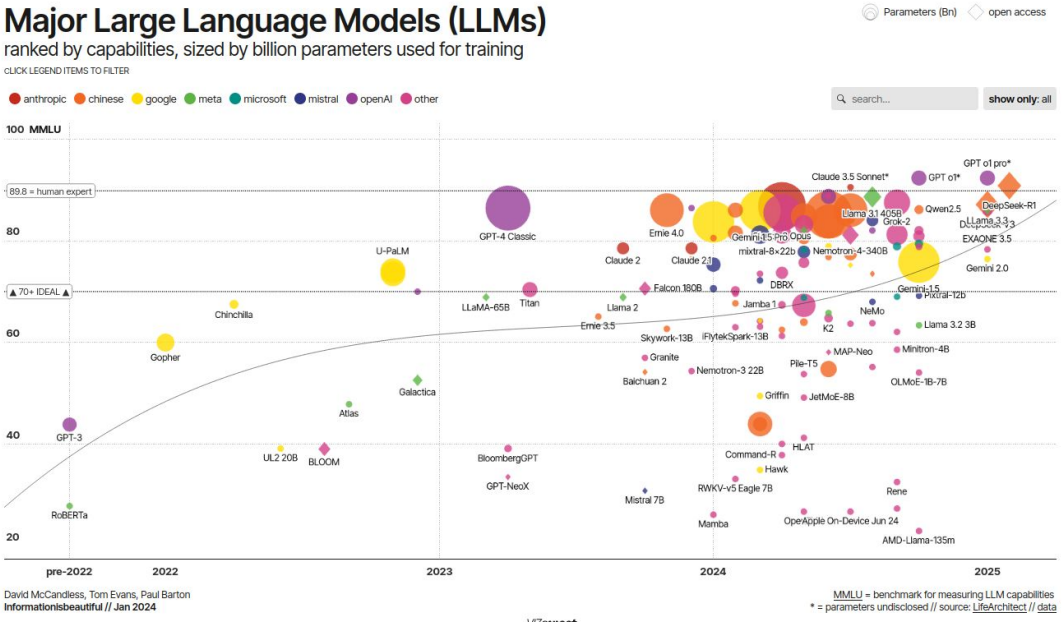




Introduction

Context and objectives

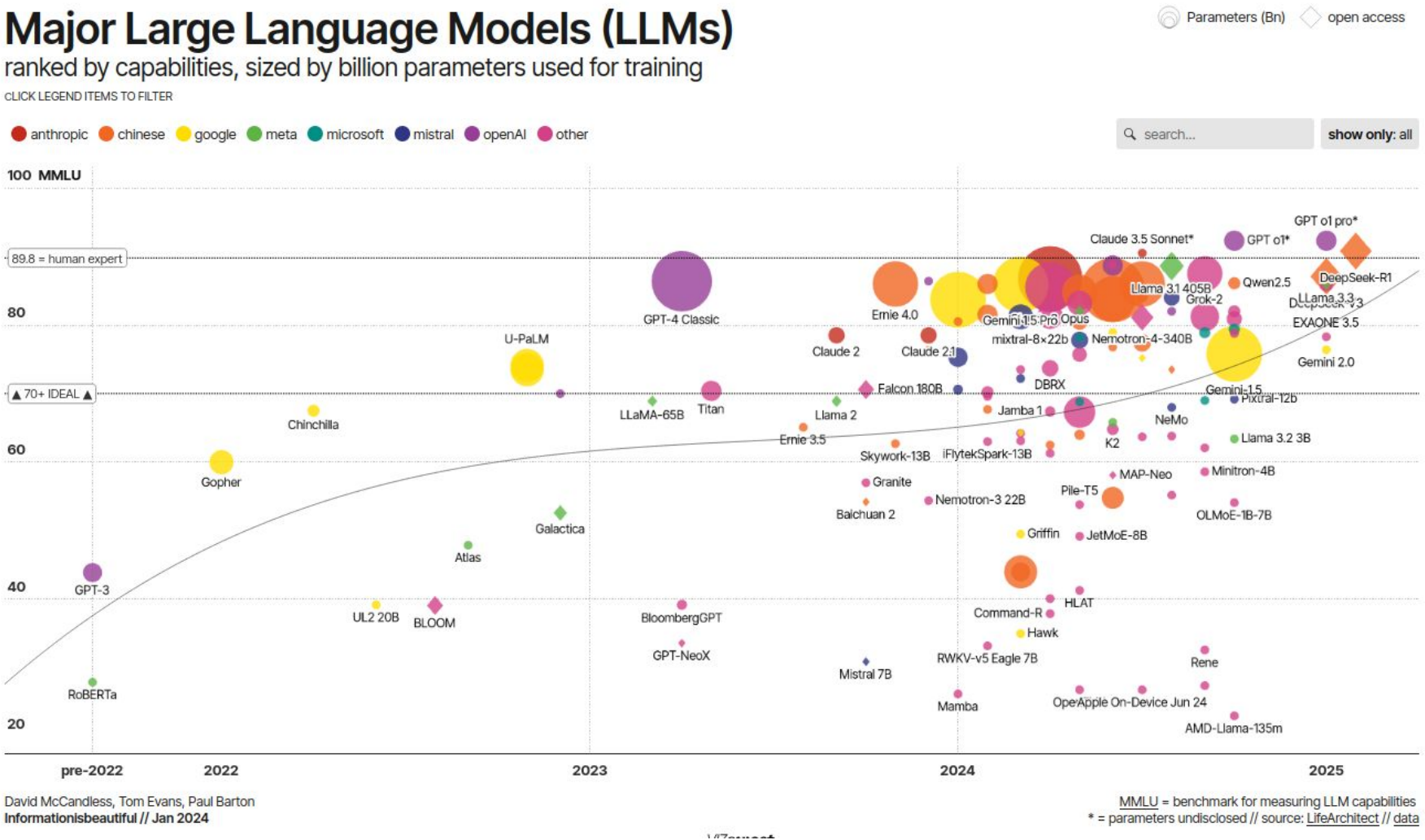
- Large Language Models (LLMs) have achieved impressive capabilities thanks to **massive datasets** and **sophisticated training techniques**.



Source: https://informationisbeautiful.net/visualizations/the-rise-of-generative-ai-large-language-models-llms-like-chatgpt/

Major Large Language Models (LLMs)

ranked by capabilities, sized by billion parameters used for training



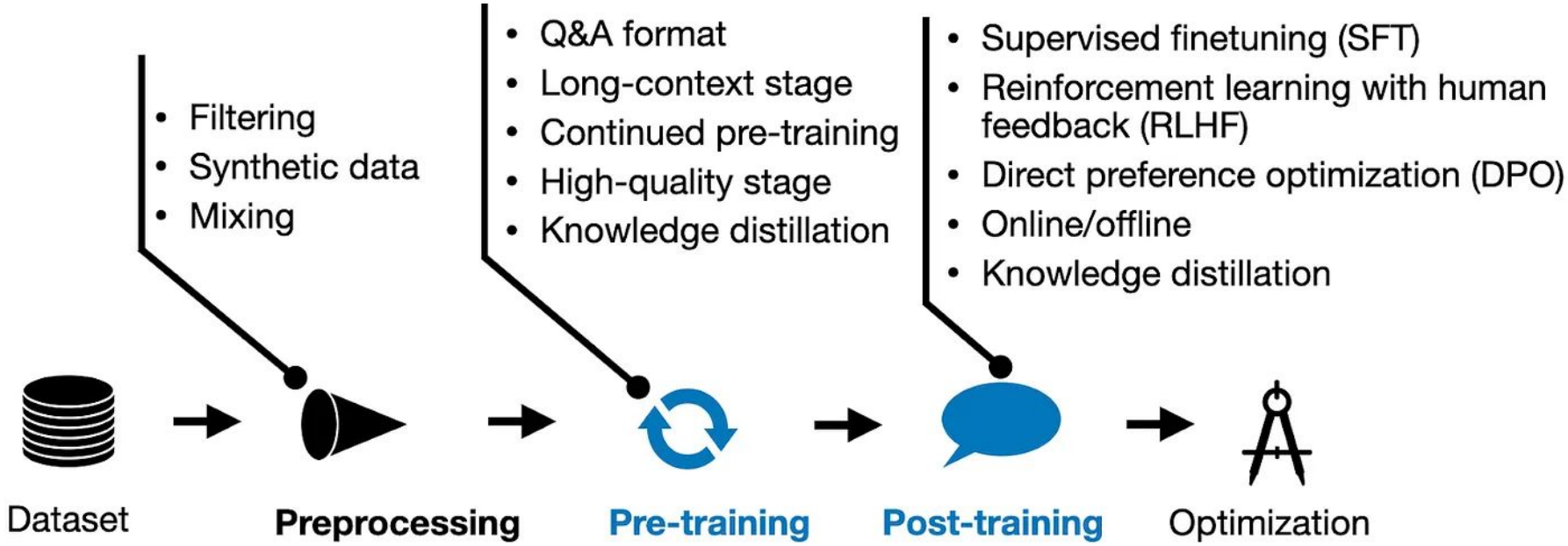
David McCandless, Tom Evans, Paul Barton
Informationisbeautiful // Jan 2024

MMLU = benchmark for measuring LLM capabilities
* = parameters undisclosed // source: LifeArchitect // data

Context and objectives

- Current LLMs are the result of complex training procedures, often detailed in lengthy reports (e.g., the LLaMA 3 paper is 92 pages long).
- In the next two sessions, we will see the training stages of modern (large) language models, from pre-training to post-training alignment, including evaluation and Parameter-Efficient Fine-Tuning.
- To illustrate current trends, we will sometimes refer to recent technical reports from:
    - LLaMA 3 (Meta, 2024),
    - Qwen (Alibaba, 2023–2025),
    - Mistral 7B (2023),
    - Claude 3 (Anthropic, 2024),
    - Google Gemini 1.5 (2024),
    - DeepSeek-VL (2024), and others.

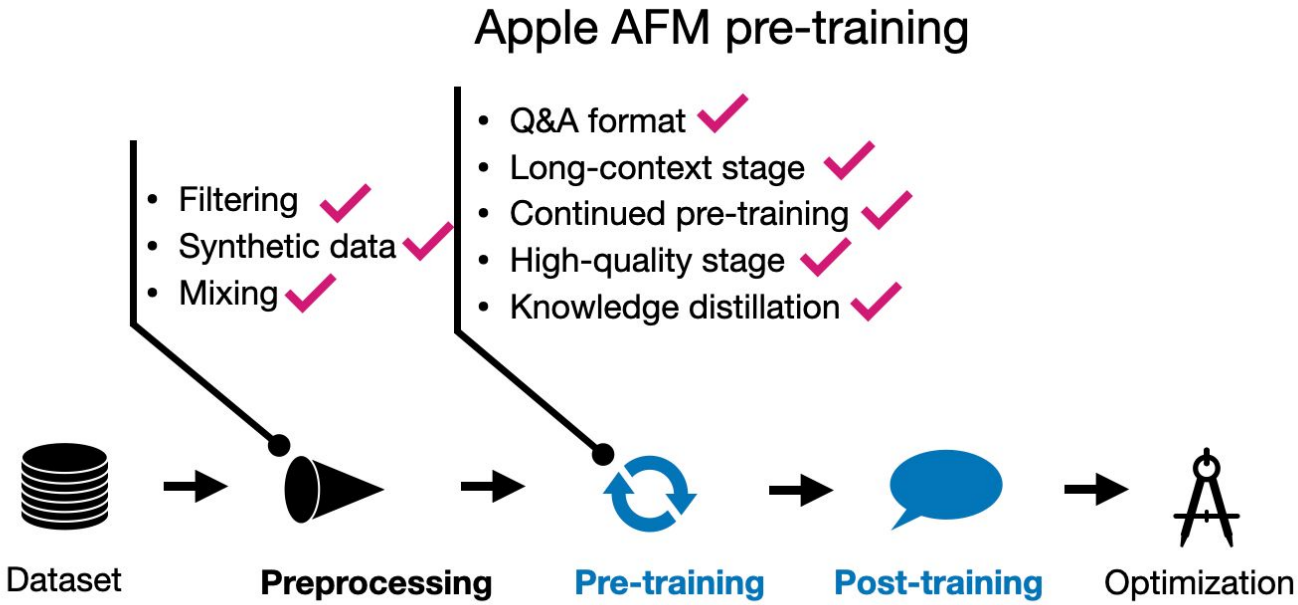
Overview



Source: https://magazine.sebastianraschka.com/p/new-llm-pre-training-and-post-training/

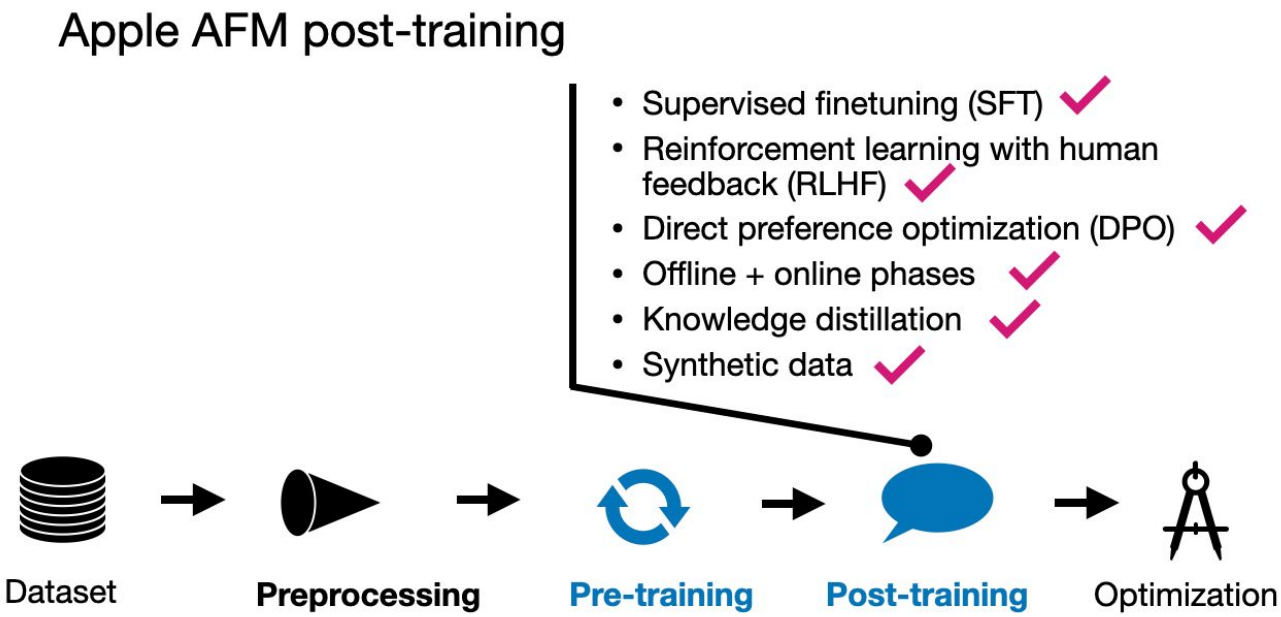
Overview

- The individual steps are not “rigid”, and you can use different combinations.



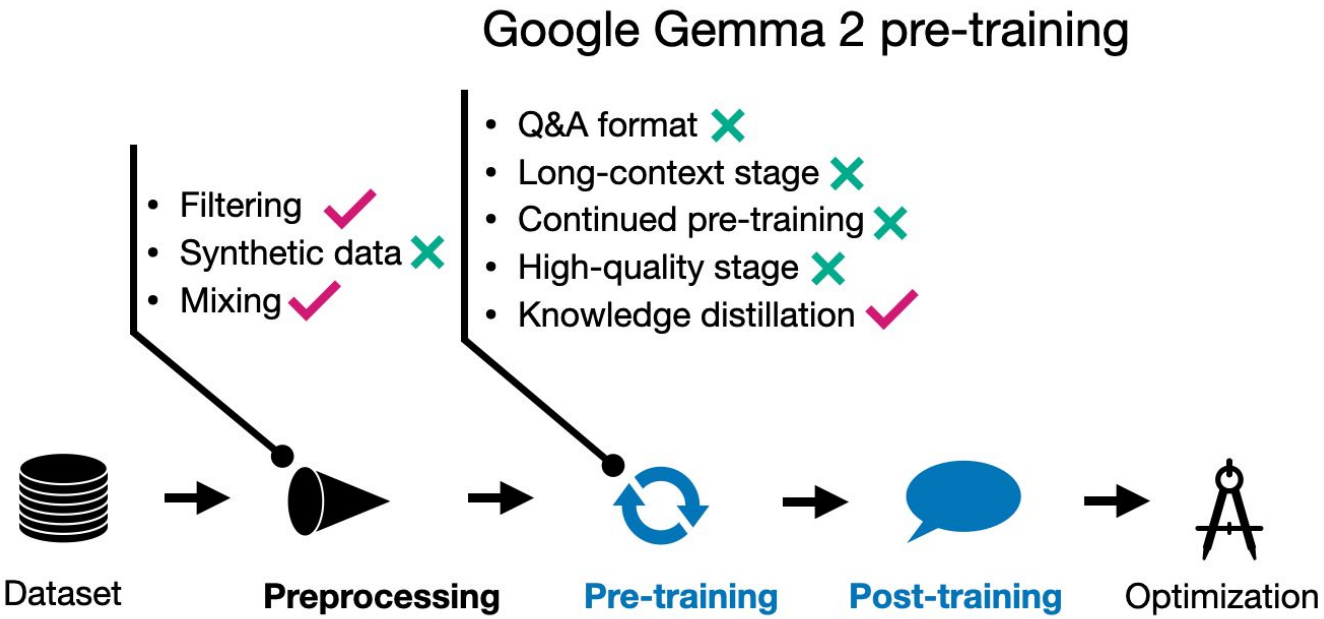
Overview

- The individual steps are not “rigid”, and you can use different combinations.



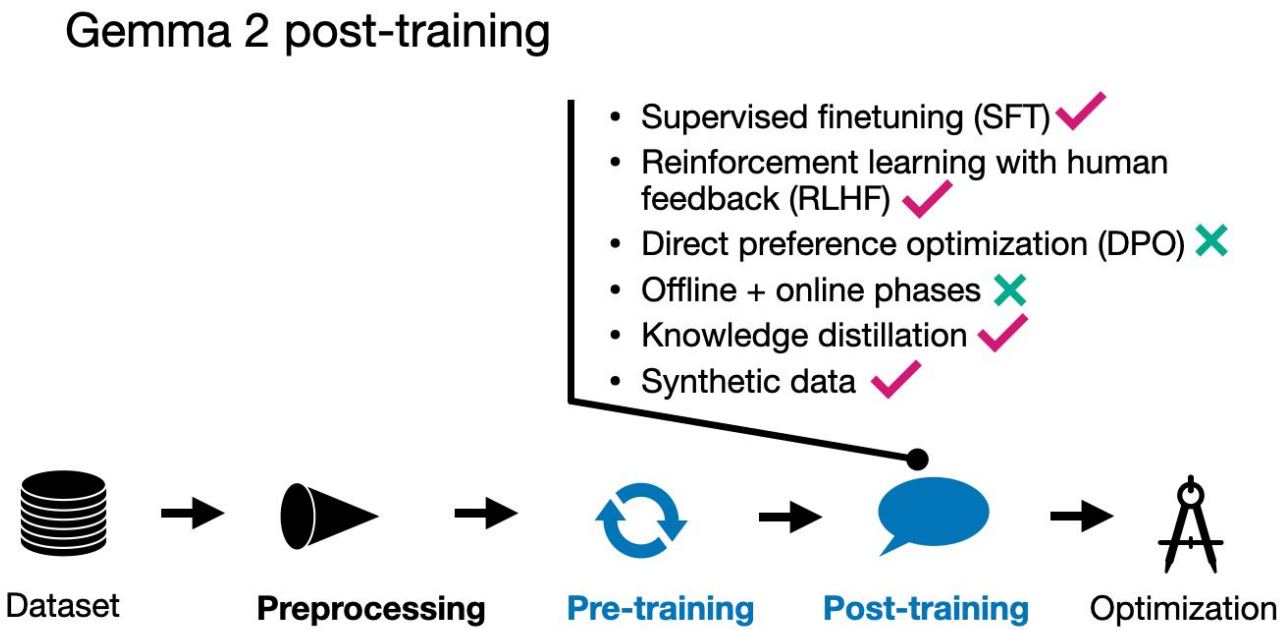
Overview

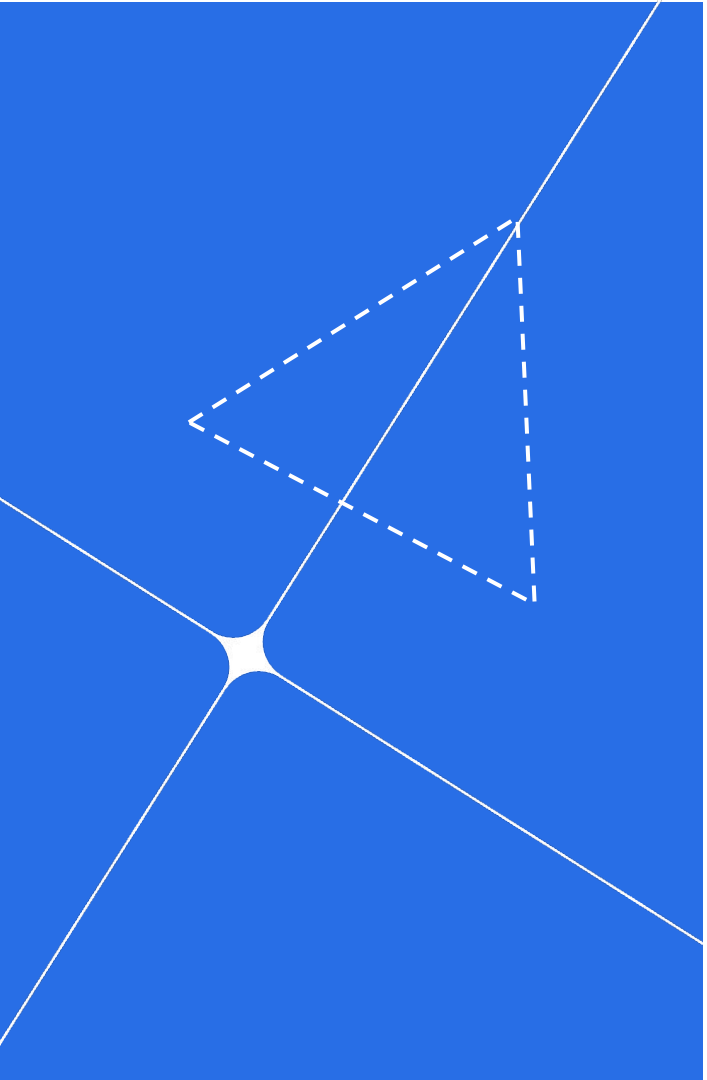
- The individual steps are not “rigid”, and you can use different combinations.



Overview

- The individual steps are not “rigid”, and you can use different combinations.





Pre-training
(and pre-processing)

Overview of pre-training

Definition:

- Pre-training consists of training a large language model on an extremely large text corpus in a self-supervised manner, by predicting masked or next words (or *tokens*).

Objective:

- Learning a general text-based representation of the world. After pre-training, an LLM is a *foundation* model capable of generating coherent text but not yet aligned with human instructions nor with specific tasks.
- The aim is to minimise the *perplexity* of a text (see *evaluation*).

Overview of pre-training

Basic architecture:

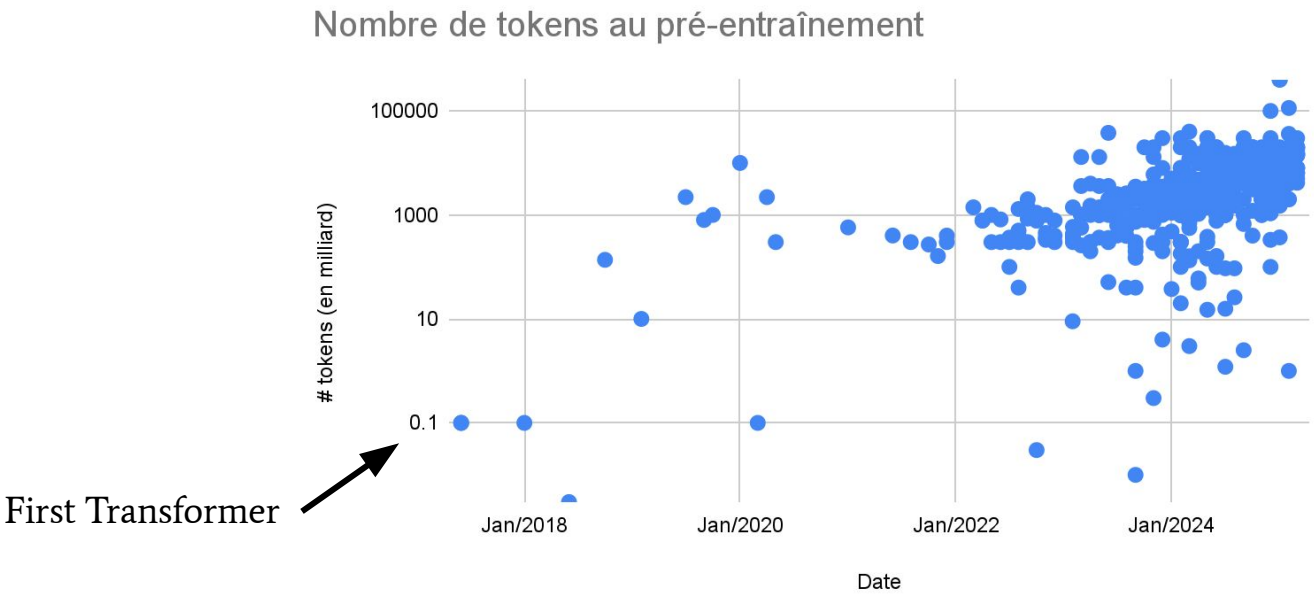
- Almost all LLMs use the **Transformer** architecture (autoregressive decoder) introduced in 2017.
- Even most recent models confirm that the strengths of LLMs come *less from architectural innovations than from a huge volume of high-quality data, careful iterative training, and scaling*.
    - Scaling → model size, inference time, (high-quality) training data.

Challenges:

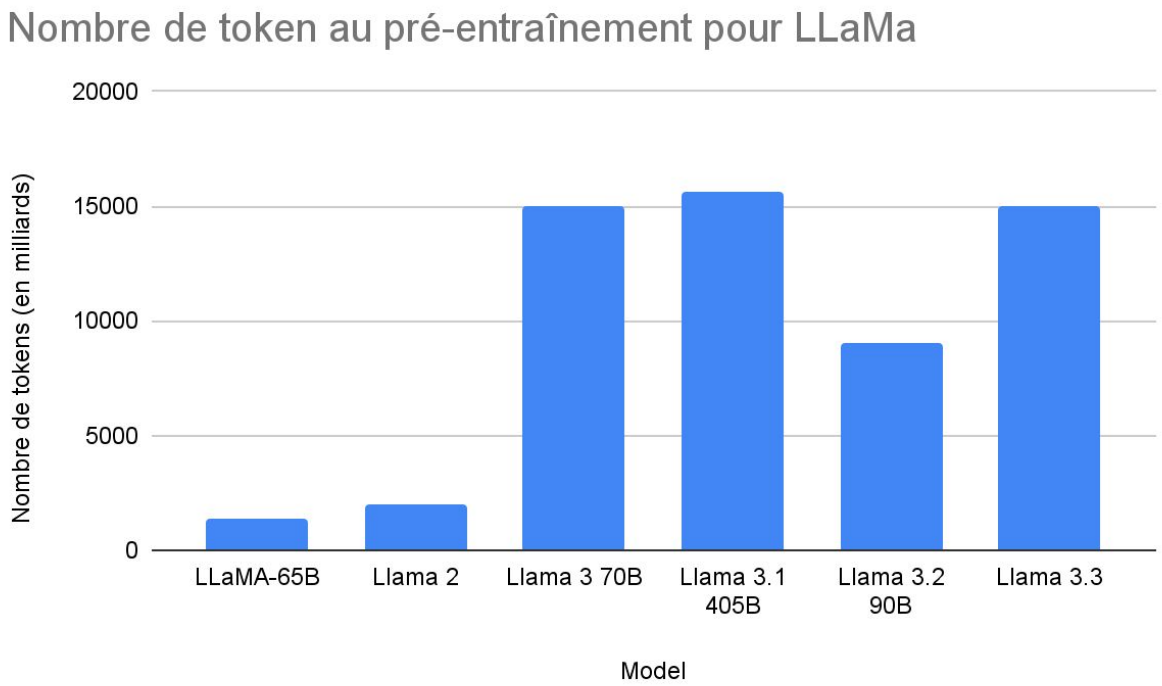
- Collecting diverse data.
- Filter noisy content.
- Manage vocabulary size and context length (e.g., the context window has increased from 2k tokens to 100k+ in the 2024 models).
- Optimise computing costs.

Web-wide corpora

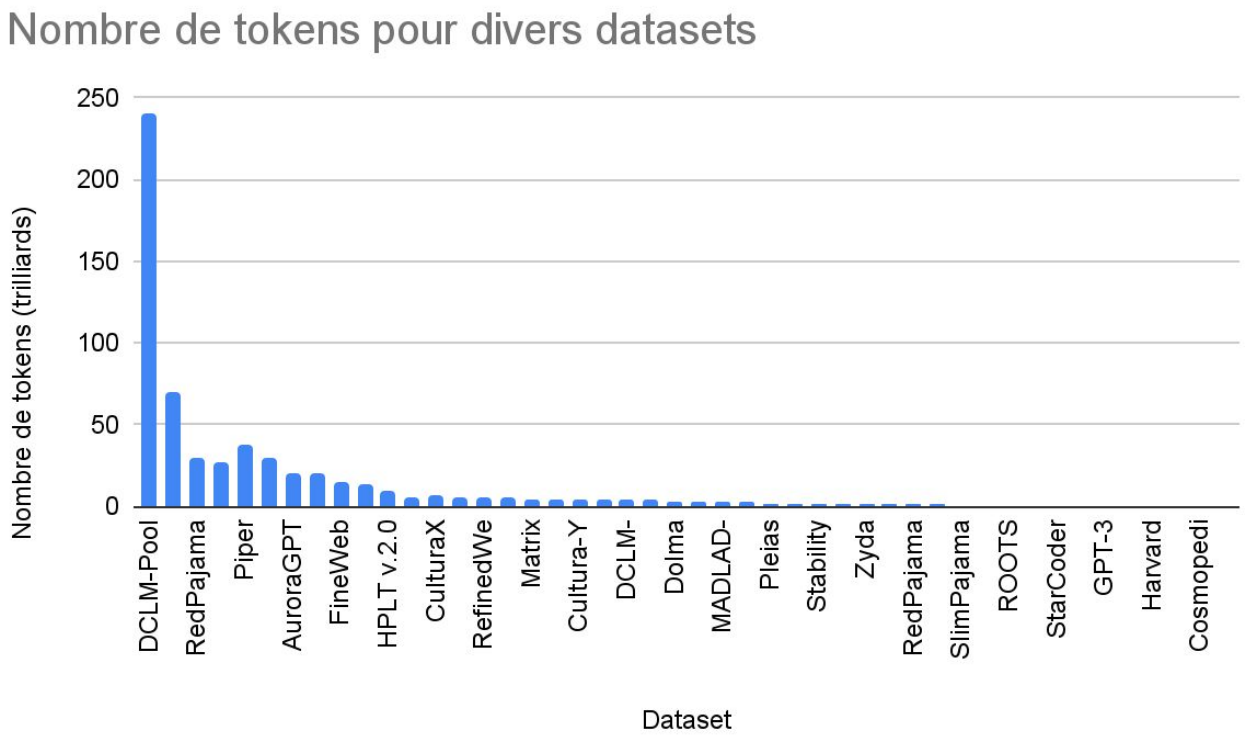
Modern LLMs are trained on colossal corpora (web text, books, scientific articles, source code, etc.).
One could basically say that these models have “seen” the *whole* internet.



Web-wide corpora



Datasets


Nombre de tokens pour divers datasets
Nombre de tokens (trilliards)
250
200
150
100
50
0
DCLM-Pool
RedPajama
Piper
AuroraGPT
FineWeb
HPLT v.2.0
CulturaX
RefinedWe
Matrix
Cultura-Y
DCLM-
Dolma
MADLAD-
Pleias
Stability
Zyda
RedPajama
SlimPajama
ROOTS
StarCoder
GPT-3
Harvard
Cosmopedi
Dataset

Linguistic and thematic diversity

- Size is not enough, a **diverse** corpus is crucial for robustness.
- Mistral 7B (2023) has been trained on a vast and diverse dataset covering multiple domains and languages, enabling it to excel at a variety of tasks despite its smaller size.
- LLaMA 3 focused on collecting high-quality web text in multiple languages and domains.
- Different data sources to have this diversity in the training data:
    - Common Crawl,
    - Wikipedia,
    - GitHub,
    - arXiv,
    - books (sometimes illegally),
    - etc.

Linguistic and thematic diversity

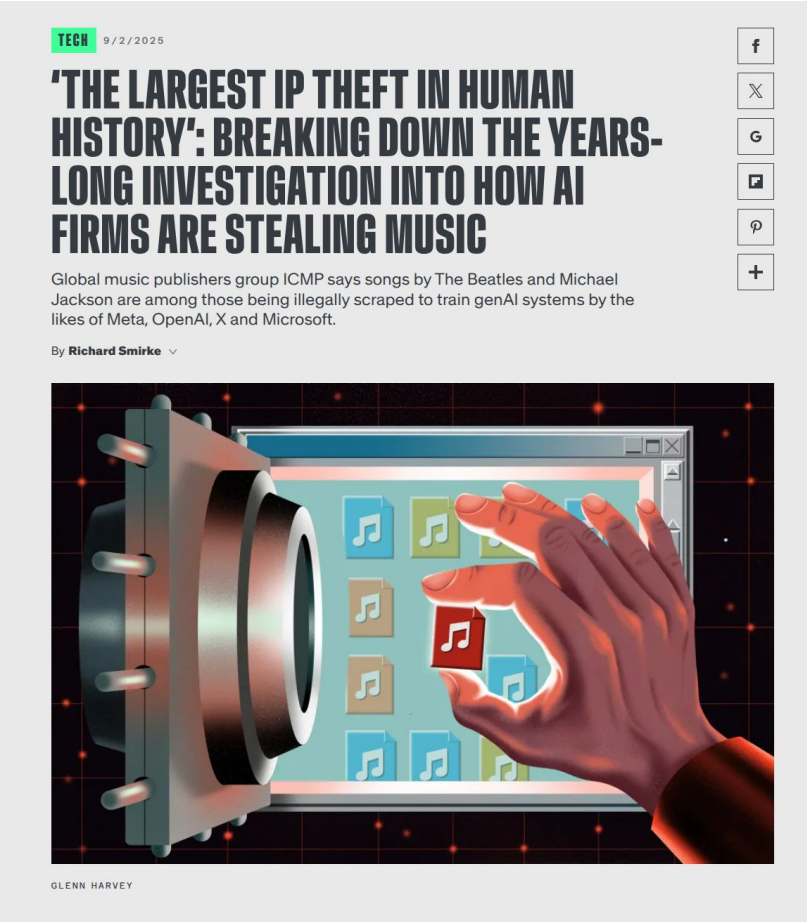
- The legality of using data for training purposes is a serious issue.

Zuckerberg approved Meta's use of 'pirated' books to train AI models, authors claim

Sarah Silverman and others file court case claiming CEO approved use of dataset despite warnings

Linguistic and thematic diversity

- The legality of using data for training purposes is a serious issue.
- For Generative AI in general, not only Language Models.



TECH 9/2/2025

‘THE LARGEST IP THEFT IN HUMAN HISTORY’: BREAKING DOWN THE YEARS-LONG INVESTIGATION INTO HOW AI FIRMS ARE STEALING MUSIC

Global music publishers group ICMP says songs by The Beatles and Michael Jackson are among those being illegally scraped to train genAI systems by the likes of Meta, OpenAI, X and Microsoft.

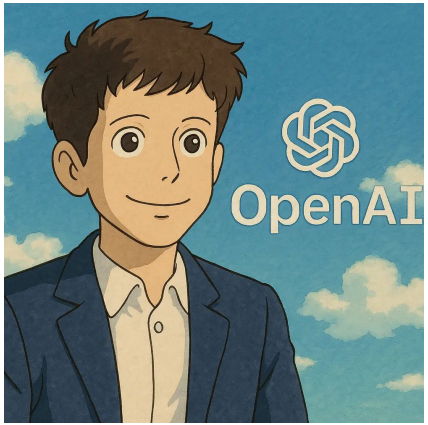
By **Richard Smirke**

GLENN HARVEY

Source: https://www.billboard.com/pro/ai-firms-steal-music-scrape-copyright-icmp-investigation

Linguistic and thematic diversity

- The legality of using data for training purposes is a serious issue.
- For Generative AI in general, not only Language Models.

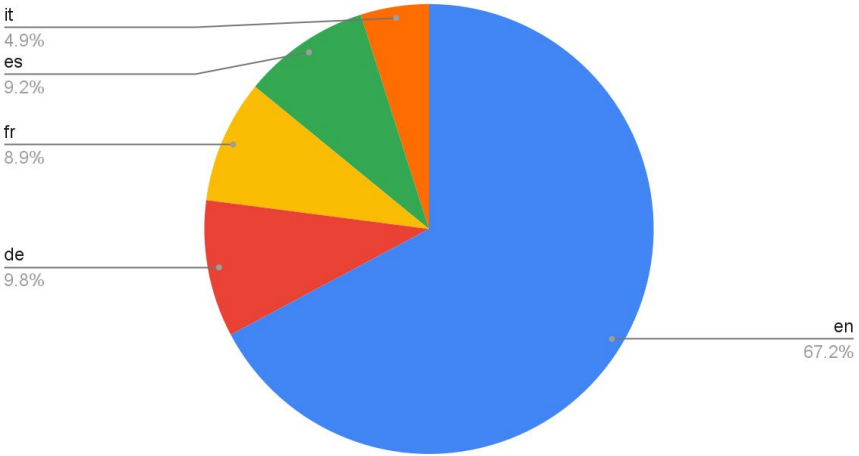


Sources (L): https://www.forbes.com/sites/torconstantino/2025/05/06/the-studio-ghibli-dilemma--copyright-in-the-age-of-generative-ai/
Sources (R): https://www.businessinsider.com/openai-studio-ghibli-image-generator-copyright-debate-sam-altman-2025-3

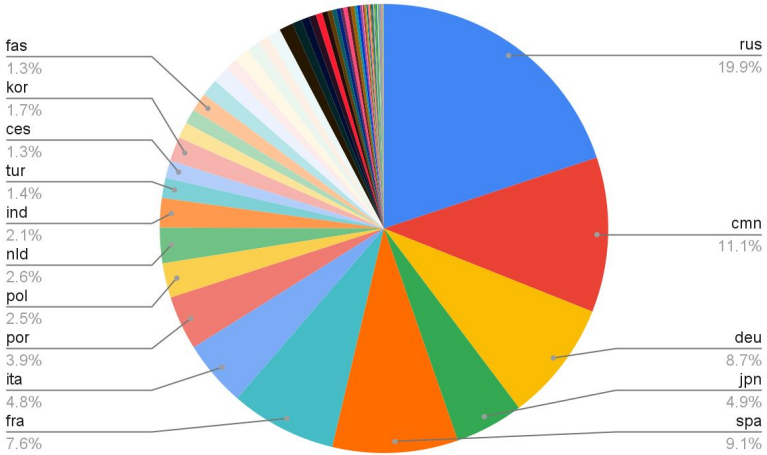
Linguistic and thematic diversity

- Diversity also means **different languages**.
- …but not all languages are equally represented in training (and evaluation) data.
- Smaller representation in data = worse performance.
    - This is particularly problematic for *low-resource* languages.

Nombre de tokens par langue dans RedPajama (LLaMa 1)



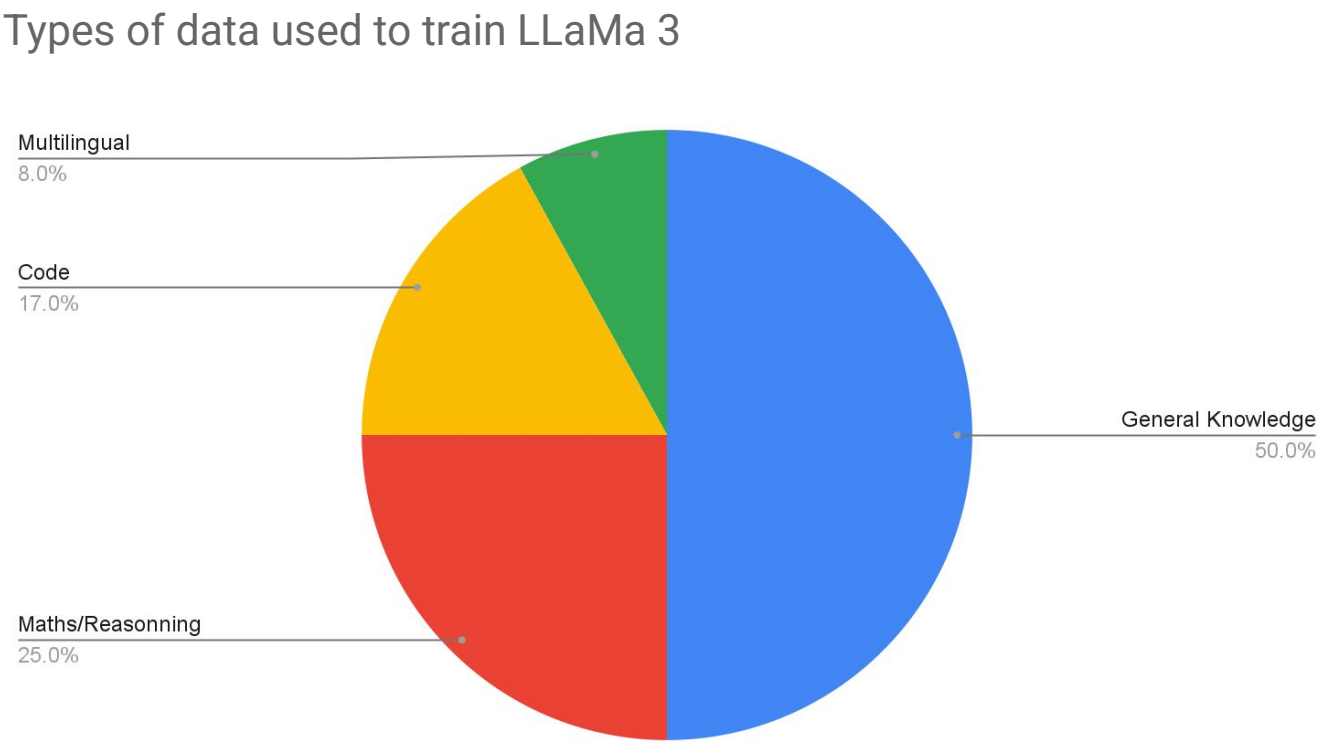
Languages in FineWeb 2 (except English)



Linguistic and thematic diversity

- In addition to generic text, teams integrate data **from specific domains** (code, math, scientific data) to equip the language model with specific skills (e.g., programming, arithmetic reasoning).
- For example, LLaMA 3 has built dedicated pipelines to extract content from web pages focused on code and mathematics.

Linguistic and thematic diversity



Data pre-processing and filtering

- **A lot** of the work.
- **Massive cleaning:** raw web data contains noise (incorrect language, spam, duplicate content, toxic content, etc.). Filtering is a crucial step. Automated filters detect and discard low-quality or irrelevant pages.
    - LLaMa 3 uses a custom HTML parser to extract high-quality text.
    - Special attention to mathematical code and formulas.
    - This works only to some extent, and additional measures are needed to e.g. block the model from generating toxic content.

Data pre-processing and filtering

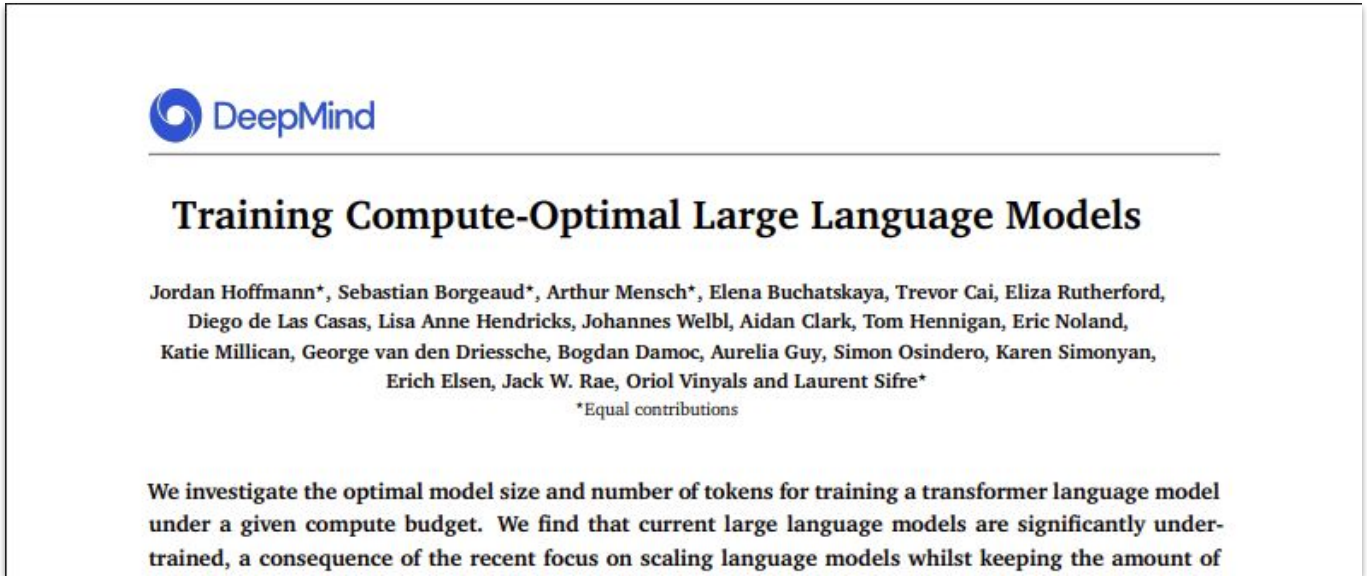
- Examples of filters:
  - **Personal and dangerous information:** LLaMa 3 has implemented classifiers to judge whether data contains sensitive or inappropriate information.
  - **Quality classifiers:** LLaMa 3 experimented with classifiers trained to evaluate the quality of text extracts and retain only tokens deemed useful. Similarly, OpenAI filtered Common Crawl using models that evaluated ‘quality’ or ‘probability of being from a book’.

Data pre-processing and filtering

- Examples of filters:
  - **Deduplication**: To avoid seeing and learning the same data, potentially from different sources, multiple times (which distorts the evaluation and can cause overfitting), large-scale deduplication algorithms are applied.
    - Example: the GPT-3 corpus was deduplicated using n-grams, and LLaMA 2 published its exclusion criteria (banned URLs, MinHash, document and line level, n-grams).
  - **Use of heuristics**: Avoid repetition of words, n-grams, inappropriate vocabulary, outlier calculations, etc.

Size of the corpus

- **Context**: Before 2022, the trend was to make models as huge as possible (e.g., GPT-3 175B), often with insufficient data. The "Chinchilla" paper changed everything by proving that for a fixed budget, you should train smaller models on more data.



DeepMind

Training Compute-Optimal Large Language Models

Jordan Hoffmann*, Sebastian Borgeaud*, Arthur Mensch*, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals and Laurent Sifre*

*Equal contributions

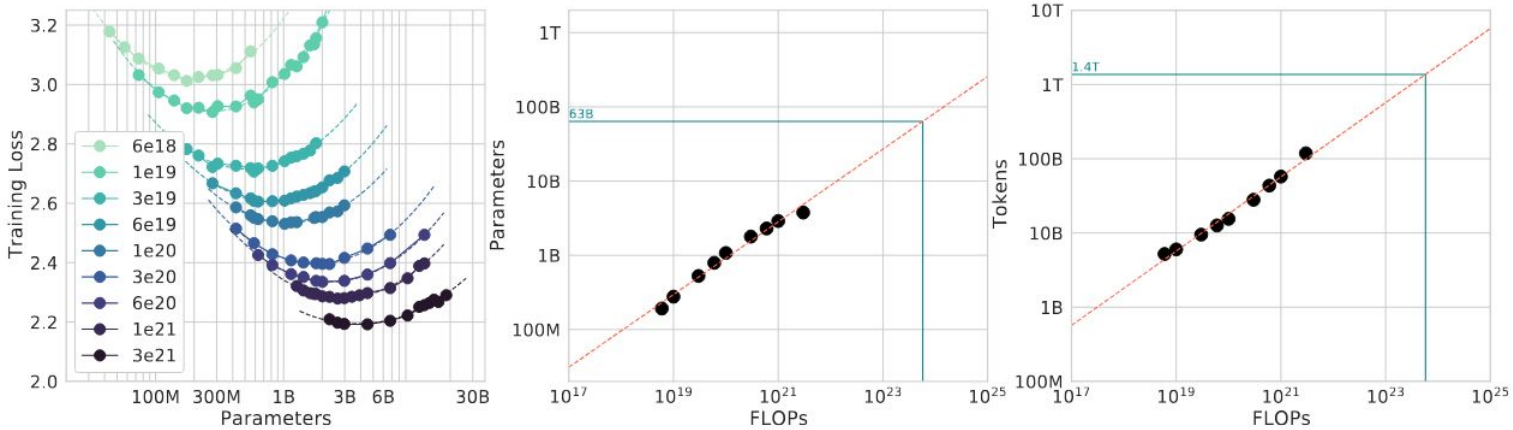
We investigate the optimal model size and number of tokens for training a transformer language model under a given compute budget. We find that current large language models are significantly under-trained, a consequence of the recent focus on scaling language models whilst keeping the amount of

Size of the corpus

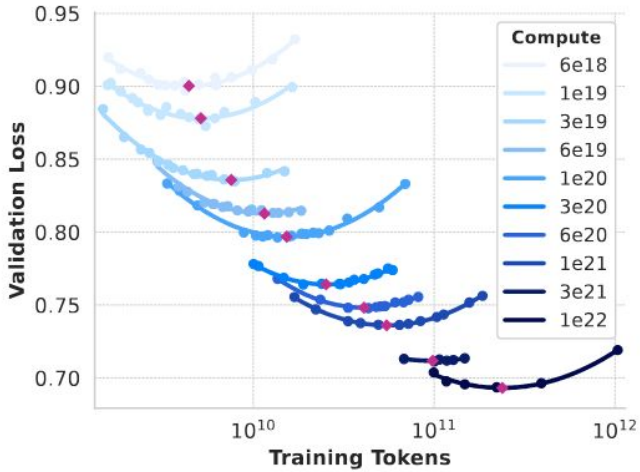
- **Scale/quality tradeoff**:
  - Finding the right balance between model size and data volume is crucial.
  - Scaling laws suggest that, given a fixed computing budget, it is better to have a smaller model that has been trained on more tokens. (E.g., the Chinchilla 70B model outperformed an under-trained 175B model by being trained for longer).
- **Rule of thumb**:
  - Chinchilla's law suggests ~20 to 30 tokens per model parameter for optimal training.
  - In practice, LLaMA 1 (65B) ~1.0T tokens (≈15 tokens/par.), LLaMA 2 (70B) ~2.0T tokens (≈30 tokens/par.).
  - The 2024 models tend to increase this ratio even further, a sign that we have not yet saturated our thirst for data.

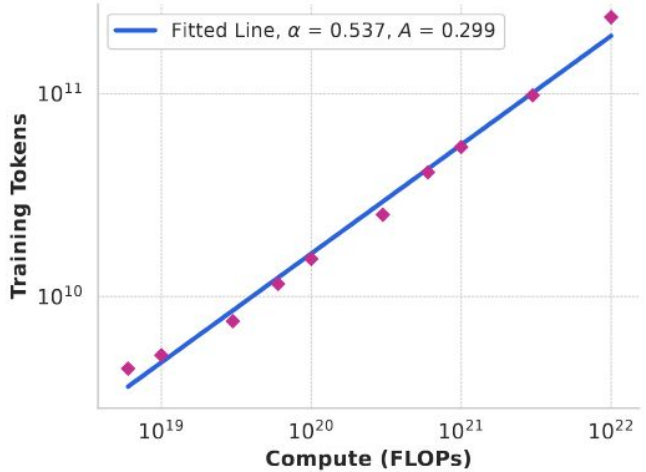
Chinchilla IsoFLOPs - 2022

For a given FLOPs budget, how can we find the best compromise between model size and number of training tokens?

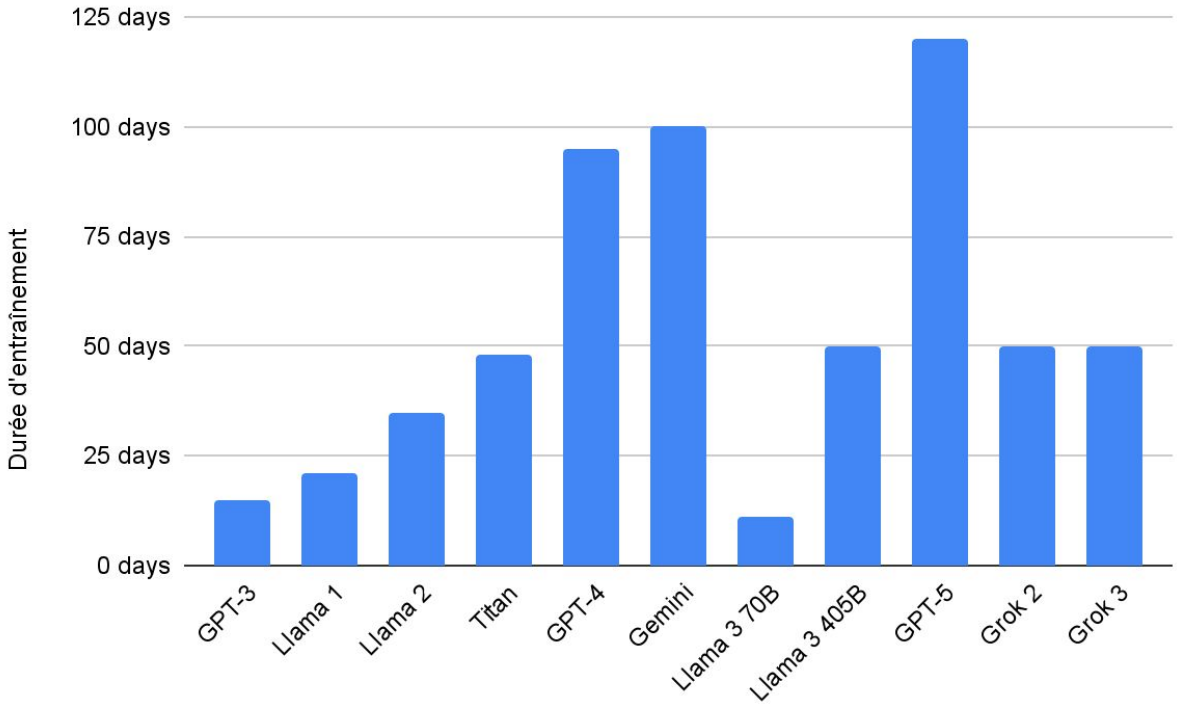


LLaMa 3 IsoFLOPs - 2024

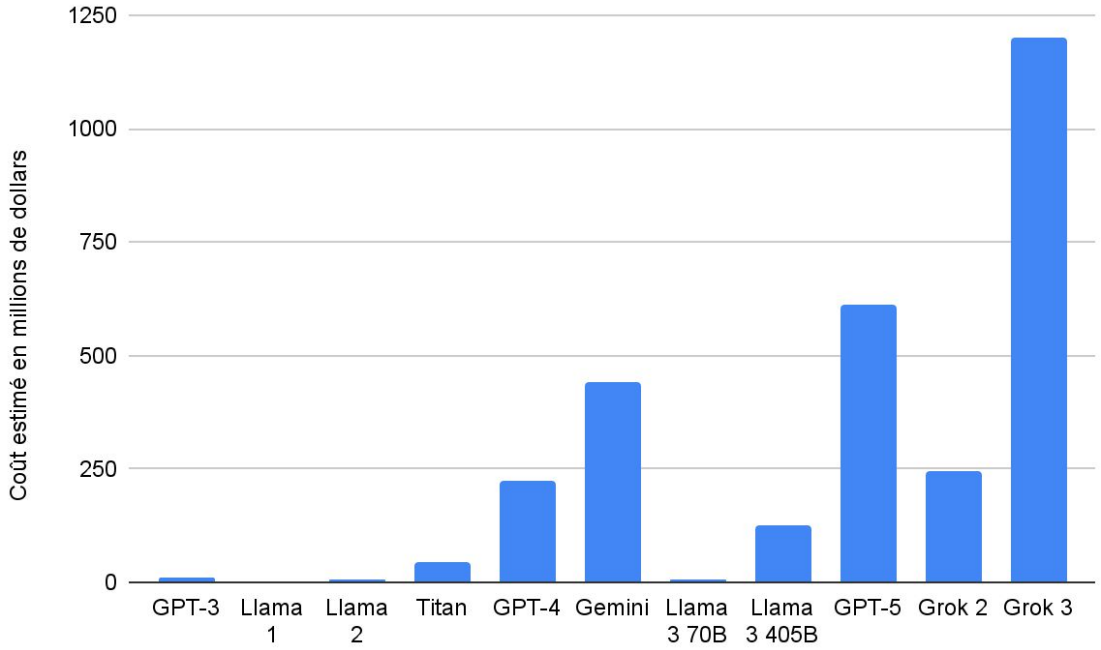




Duration of training



Cost of training (estimated)



Tricks to save time (and money)

- Pre-training an LLM costs millions of dollars in GPU computing.
- Several tricks can be performed to save time and money.
  - **Mixed Precision** (FP16/BF16): We don't need 32 decimal places of precision for every number. Using 16-bit (half precision) cuts memory usage in half and speeds up math.
  - **FlashAttention**: A hardware-aware algorithm. It reduces the number of times the GPU has to read/write to its slow memory (HBM), making attention calculation much faster.

Other tricks used during training

- **Scheduling the data path:** Training is generally performed in epochs on the corpus.
  - Some teams use **implicit curriculum learning**: for example, starting training with shorter sequences and then gradually increasing the length of the context.
    - Definition *curriculum learning*: a machine learning technique that involves training a model on a sequence of tasks or examples of increasing difficulty, starting with simpler ones and gradually progressing to more complex ones.
  - Others apply **non-uniform sampling**: LLaMA 3 oversampled high-quality data (code, math) at the end of pre-training to boost performance in those domains.
- **Learning rate**: A high learning rate is used at the beginning (after warm-up), followed by a gradual decrease, often using cosine decay. LLaMA 3 used cosine annealing at the end, in conjunction with upsampling certain types of data.
  - Training stability is a challenge.
  - With increasingly longer contexts (e.g., 100k tokens, such as Gemini 1.5 or Claude 2), the effective duration of a gradient step increases and also requires adjusting these schedules.

Other tricks used during training

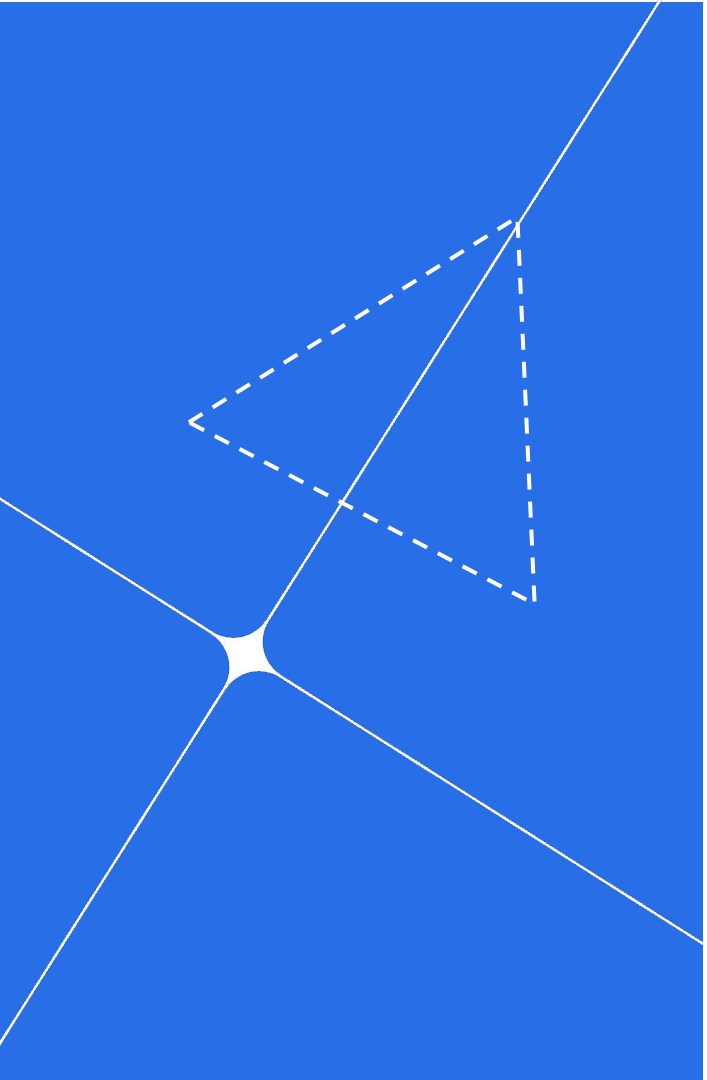
- **Data augmentation**: We already have a lot of text, but there are a few approaches for augmenting (i.e. oversampling) the pre-training dataset with synthetic data:
  - generative self-play: having a smaller model produce text to increase the corpus.
  - machine translation to add languages.
- But these techniques are marginal for pre-training (more commonly used in post-training).

Other tricks used during training

- **Long context management**: To increase the context window without exploding the cost of attention (which is quadratic), some tricks are used:
  - Mistral 7B uses sliding window attention to process arbitrarily long sequences, limiting attention to a local sliding window. This allows it to handle long inputs at a reduced cost.
  - Other models use advanced positional embeddings (RoPE, ALiBi) to extrapolate to long sequences.

Other tricks used during training

- **Optimized tokenization**:
  - The choice of tokenizer (BPE, Unigram) and vocabulary (e.g., 32k vs. 100k tokens) impacts training.
  - Multilingual models have larger vocabularies.
  - Code models incorporate special tokens for programming symbols.
  - Qwen uses ~150k tokens, including special control tokens.
  - LLaMa = 128k tokens = 100k + 28k for non-English support.
    - Tokenization is a challenge for multilinguality, since tokenizers are often optimized for English.



Evaluation of pre-training

Why evaluating?

- To measure the capabilities of an LLM and guide its improvement.
- Since LLMs are versatile, we cannot rely on a single metric, and multiple evaluation criteria are used.
- Types of evaluation:
  - **Internal metrics** such as perplexity on a secret text corpus (indicator of the model's quality as a language model);
  - **External benchmarks** on standardized tasks (Q&A, reasoning, code, etc.), allowing models to be compared with each other;
  - **Human evaluation or** evaluation **via other models** on aspects such as response preference and usefulness in conversation;
  - **Safety checks** and **other evaluations**: bias, toxicity, hallucinations, robustness to rephrasing.
- We will focus now on the techniques used on models after pre-training, in the next lecture we will look at additional evaluation techniques.

Why evaluating?

- **Pre- vs. post-training evaluation:** a pre-trained (unaligned) model is often evaluated using perplexity and a few closed tasks, whereas a conversational model must be evaluated on the quality of its free responses. This sometimes requires chatbot arenas or qualitative ratings.
- **Challenges**:
  - Ensure that the evaluation is fair and unbiased (avoid contamination where the model has seen the test responses during training).
  - Cover a wide range of areas (to avoid over-optimizing on a few popular benchmarks).
  - Take ethical criteria into account (a technically high-performing model may still be unusable if it generates biased or dangerous responses).

Perplexity and linguistic quality

- **Perplexity** is a basic metric for language models.
- It **measures how well the model predicts text it has not seen** (the lower the perplexity, the better the model).
- For example, DeepMind's RETRO 7B achieves a perplexity comparable to GPT-3 175B on The Pile using a text database, despite having 25 times fewer parameters.

$$perplexité(p) = 2^{entropy(p)}$$

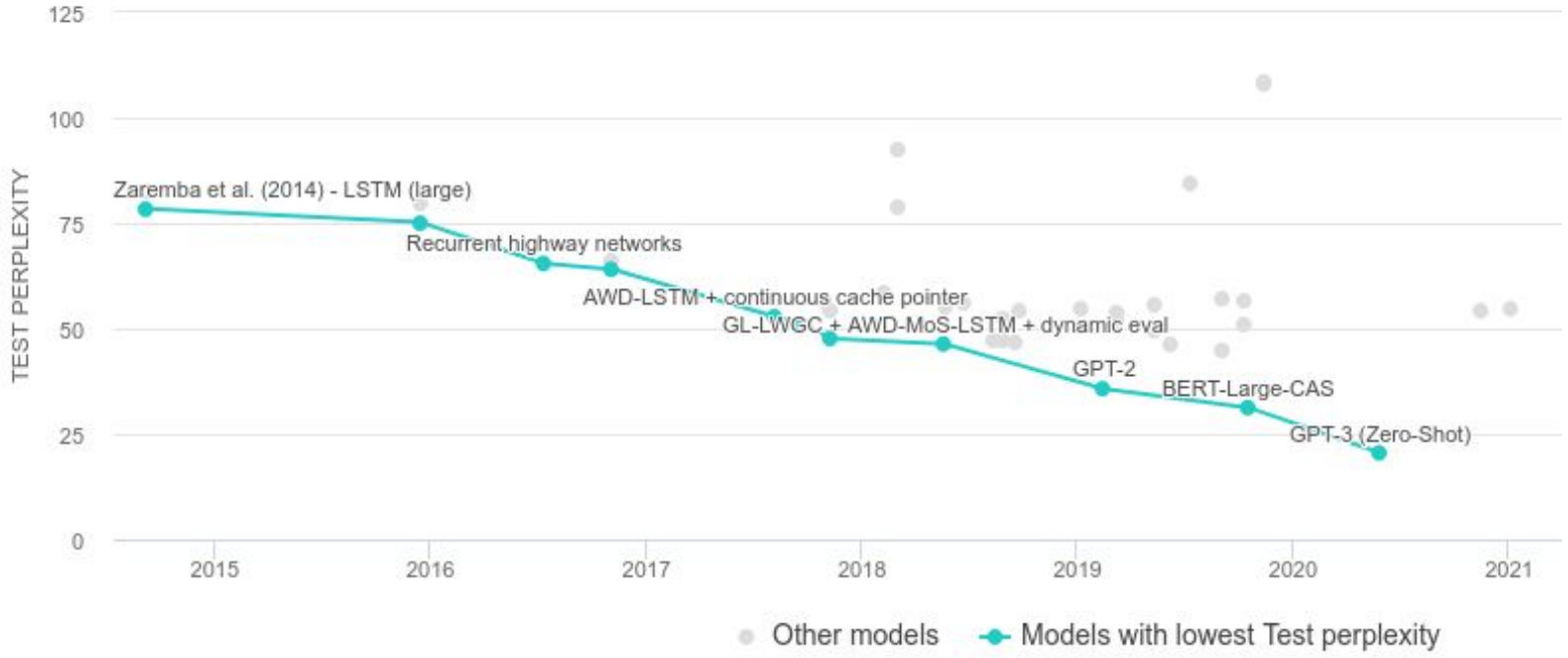
$$perplexity_p(x_1, \ldots, x_n) = exp(-\frac{1}{n}\sum_{i=1}^{t} log(p(x_i|x_{<i})))$$

$$perplexity_p(x_1, \ldots, x_n) = \frac{1}{P(x_1,\ldots,x_n)^{1/n}}$$

Perplexity and linguistic quality

- **Interpretation**: level of surprise upon seeing a sequence (in other words, how unexpected it was).
  - perplexity = 10 → the model is as surprised on average as if it had to choose between 10 options for each word (another way to see this: “as confused as if it had to roll a 10-sided dice”).
  - perplexity = 50 → model is “more surprised”, thus less good.

Perplexity and linguistic quality



Source: https://paperswithcode.com/sota/language-modelling-on-penn-treebank-word

Perplexity and linguistic quality

Utility:

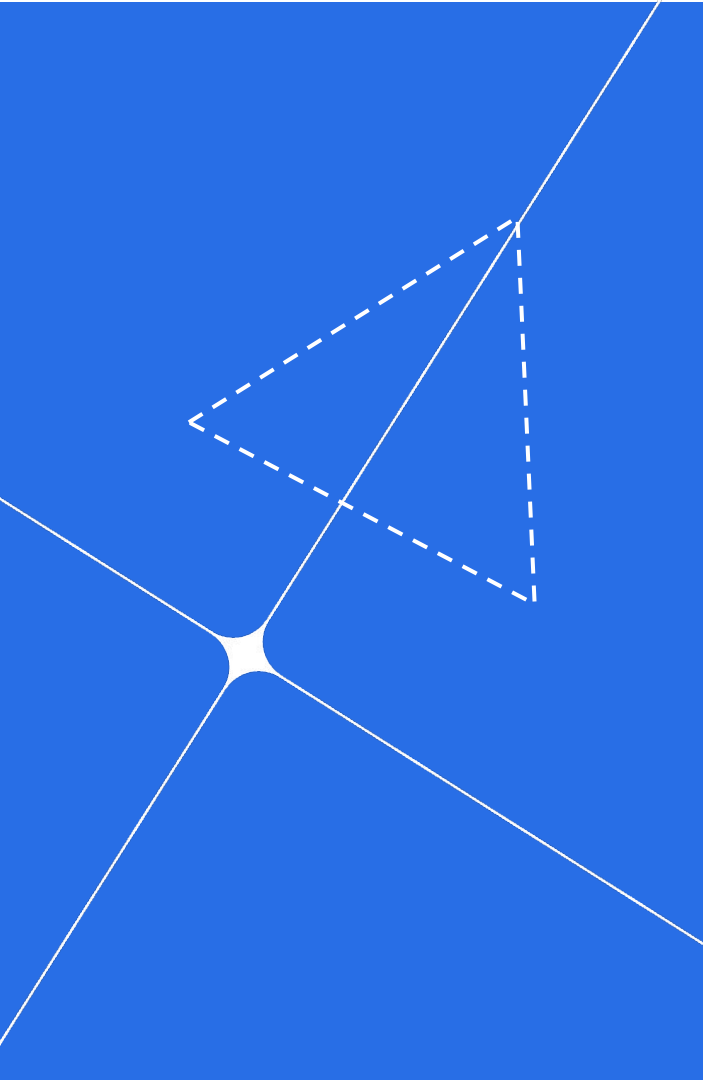
- Perplexity is mainly used to evaluate pre-training. It is calculated on a hidden test set (e.g., unseen Wikipedia pages).
- A decrease in perplexity from one model generation to the next indicates progress in overall linguistic ability.

Limites:

- Low perplexity does not guarantee that the model will perform well on specific tasks or be aligned.
  - For example, GPT-3 had much better perplexity than GPT-2, but still produced unfiltered responses or responses that were off-topic.
- Over-optimizing perplexity can sacrifice other aspects (such as diversity).

Other internal metrics

- Entropy rate.
- Vocabulary coverage.
- Loss in specific domains (e.g., perplexity can be calculated on code to estimate programming abilities).
- Manual evaluation of “bulk” completion examples to ensure that the model does not deviate.



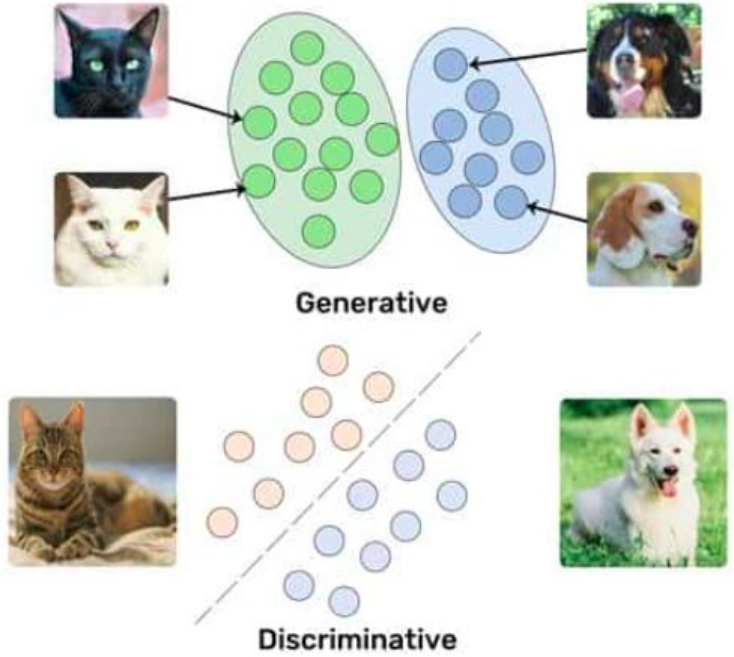
Application:
Fine-tuning for Classification

From text generation to discrimination

- LLMs are great at generating texts (poetry, chatting, …) but in many applications we often just need a specific answer:
  - Is this email Spam or Not? ← spam detection
  - Is this contract risky (0-5)?; ← risk classification
  - Is this a positive review? ← sentiment analysis
  - What does this document represent? ← topic modelling
- We could ask a text generation model to answer with specific words (e.g., {'spam', 'not spam'}), but this is inefficient and harder to evaluate programmatically (synonyms, rephrasing, …).
  - ➢ We can *teach* the models to output a probability distribution over fixed classes.
- **Key Terminology**:
  - Generative: Modeling $P(x)$ or $P(x_{next} \mid x_{prev})$.
  - Discriminative: Modeling $P(y|x)$ where y is a discrete label.

From text generation to discrimination

- A (multimodal) example.
- **Generative Model**: receives an image, generate a description (race, colour…).
- **Discriminative Model**: receives an image, returns a binary classification (cat vs. dog)



From text generation to discrimination

- Some textual examples.
- Spam detection.
  - Generative: receives an email, produces a text stating whether it is spam or not.
  - Discriminative: produces a binary classification (potentially with a probability score).
- Sentiment analysis
  - Generative: receives a movie review, and produces a textual feedback about it.
  - Discriminative: produces a score on a predefined range (e.g. 1-5).

Classification head

- We can teach the models to output probability distributions over fixed classes by using a **classification head**.
- **Mechanism**:
  - Feed input text (tokens) into the Base Model.
  - Extract the **Last Hidden State** of the final token (often the [EOS] token or a specific [CLS] token, depends on the architecture).
  - **Discard** the massive Vocabulary Projection Layer ($d_{model} \rightarrow$ 50k vocab size).
  - **Attach** a small Linear Layer ($d_{model} \rightarrow$ K classes).
  - Apply **Softmax** (for multi-class) or **Sigmoid** (for binary).
- We use the last hidden state because it contains the "semantic summary" of the sentence; the linear layer maps that summary to your specific labels.

Pooling strategies

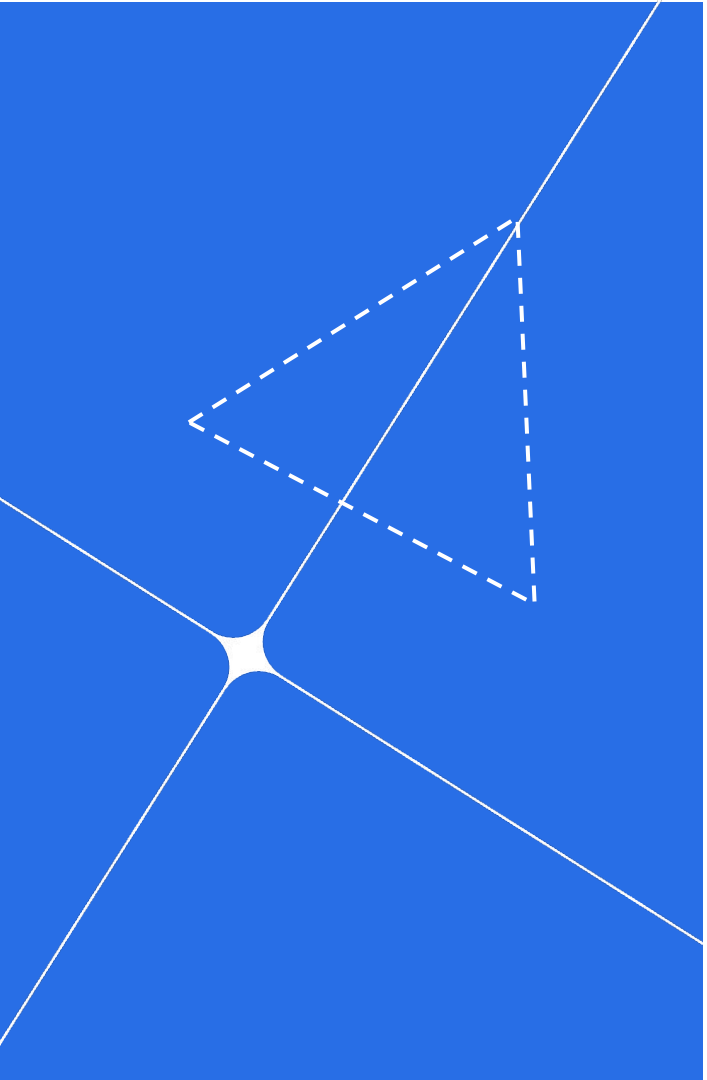
- **Issue**: The model outputs a vector for every token. Which one represents the "whole sentence"?
- **Strategies**:
  - **Last Token / EOS**: Standard for causal LLMs (GPT style). The last token has "seen" everything before it.
  - **CLS Token**: Standard for encoder-only (BERT style), but can be trained into decoders.
  - **Mean Pooling**: Averaging all token vectors (good for semantic similarity/embeddings).

The fine-tuning process

- The classification head is initialised randomly, we have to train it: this is called **fine-tuning**.
- **Weights**:
  - We initialize with the Base Model weights (pre-trained knowledge).
  - The Classification Head is initialized randomly.
- We use a different **loss function**:
  - **Pre-training**: Cross-Entropy over 50k vocabulary (predict next word).
  - **Classification**: Cross-Entropy over *N* classes (predict label).
  - **Regression**: Mean Squared Error (MSE) (predict a score, e.g., 1.5 stars).
- **Note**: This updates the entire model (unless we use PEFT/LoRA, seen in next lecture).

Today's practical session

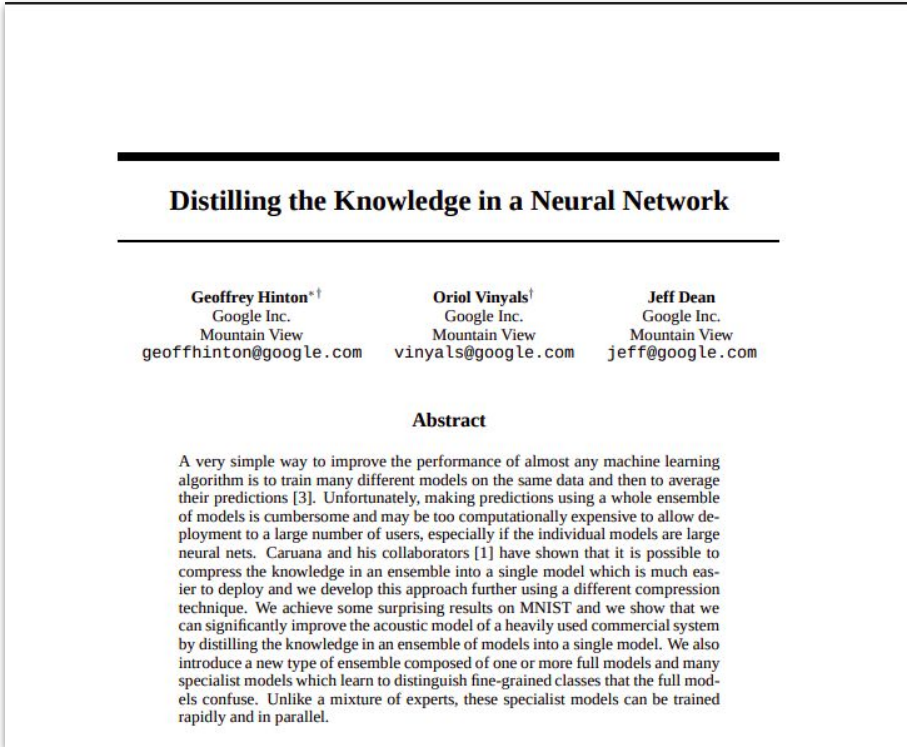
- Later today, in the practical session (TP), you will have the opportunity to work on a fine-tuning for classification problem, taking a pretrained GPT-2 model and fine-tuning it for spam detection.



Knowledge Distillation

The Inference bottleneck

- Let’s say that you have a 70B parameter model that classifies legal documents perfectly.
- However:
  - Latency: It takes 2 seconds per document.
  - Cost: You would need an A100 GPU to run it.
  - Requirement: You need to process 1 million documents a day, and only have access to a CPU.
- You cannot use that model, in this setting.
- The Solution: Knowledge Distillation. Compressing the "Teacher's" intellect into a "Student."



Distilling the Knowledge in a Neural Network

Geoffrey Hinton[*†] Google Inc. Mountain View geoffhinton@google.com

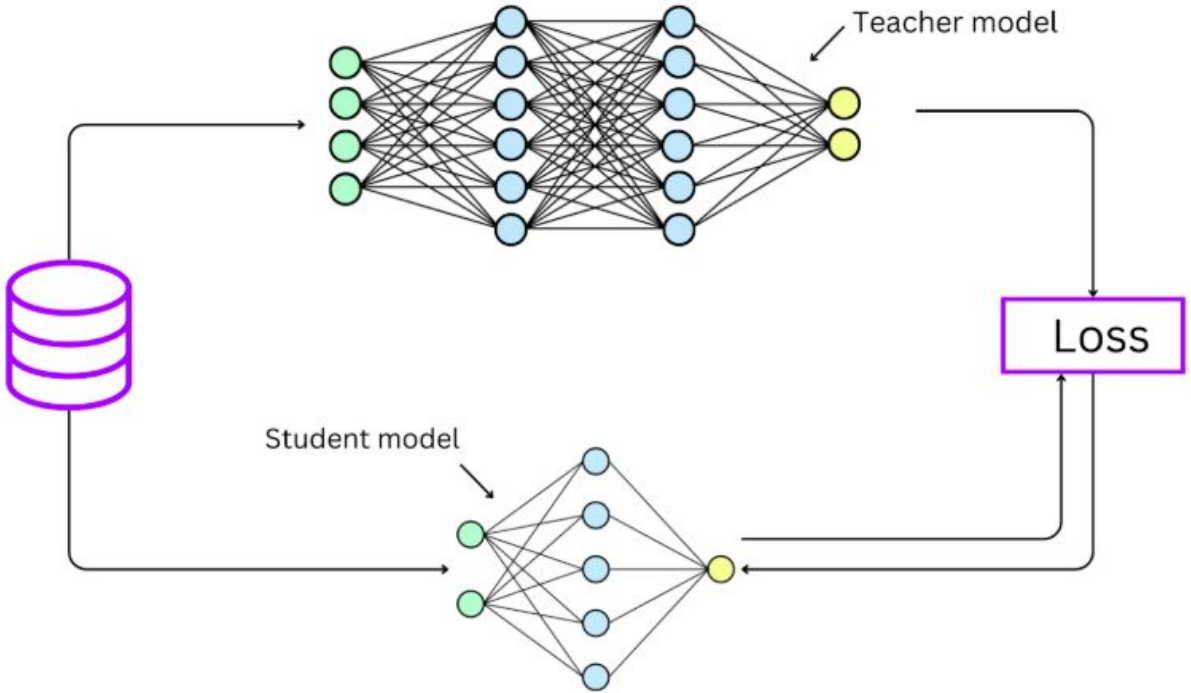
Oriol Vinyals[†] Google Inc. Mountain View vinyals@google.com

Jeff Dean Google Inc. Mountain View jeff@google.com

Abstract

A very simple way to improve the performance of almost any machine learning algorithm is to train many different models on the same data and then to average their predictions [3]. Unfortunately, making predictions using a whole ensemble of models is cumbersome and may be too computationally expensive to allow deployment to a large number of users, especially if the individual models are large neural nets. Caruana and his collaborators [1] have shown that it is possible to compress the knowledge in an ensemble into a single model which is much easier to deploy and we develop this approach further using a different compression technique. We achieve some surprising results on MNIST and we show that we can significantly improve the acoustic model of a heavily used commercial system by distilling the knowledge in an ensemble of models into a single model. We also introduce a new type of ensemble composed of one or more full models and many specialist models which learn to distinguish fine-grained classes that the full models confuse. Unlike a mixture of experts, these specialist models can be trained rapidly and in parallel.

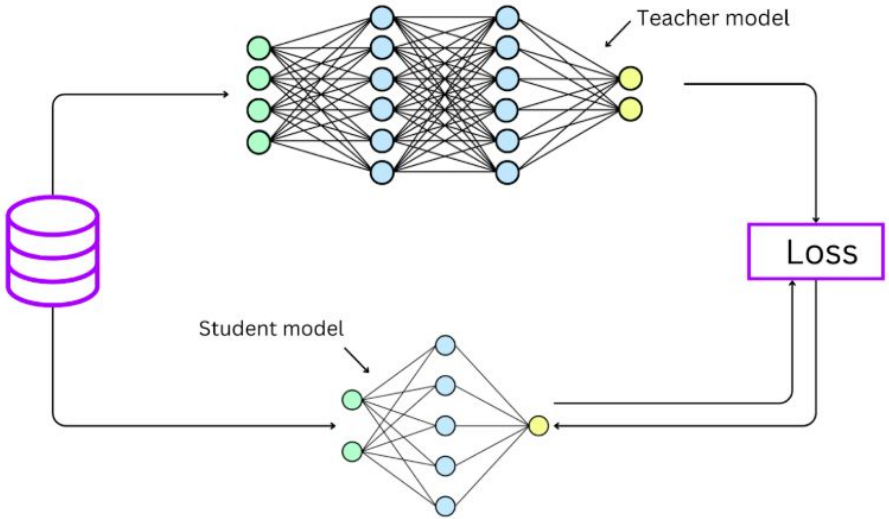
Teacher-Student Architecture



Source

Teacher-Student Architecture

- Roles:
  - **Teacher**: Large, accurate, slow (e.g., GPT-4, Llama-70B).
  - **Student**: Small, fast, initially dumb (e.g., TinyLlama, DistilBERT).
  - **Goal**: Student mimics the Teacher, not just the Ground Truth.



“Dark Knowledge" & Soft Targets

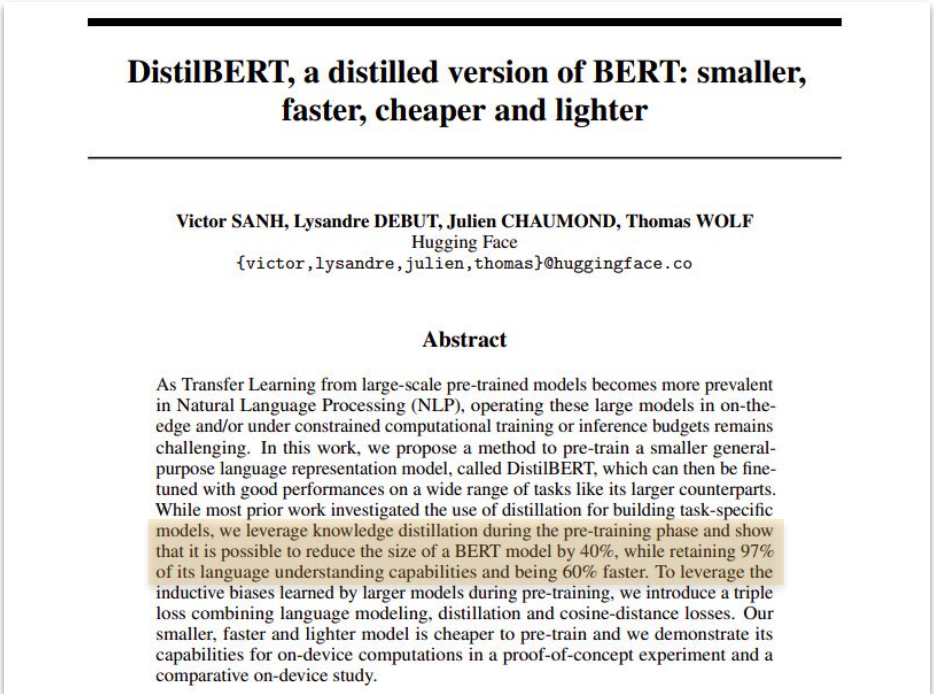
- Hinton’s Insight (2015): **The "wrong" answers contain information**.
- An example (from image classification):
  - Image: A Golden Retriever.
  - Hard Label (Ground Truth): [1, 0, 0, 0] (Dog, Cat, Car, Boat).
  - Teacher's Logits: [0.90, 0.09, 0.009, 0.001].
  - The Teacher tells the Student: "It's definitely a dog, but if it wasn't a dog, it would look like a cat, not a boat."
- This relationship (Dog ~ Cat != Boat) is the *Dark Knowledge* or "structural understanding" of the world.
- Temperature (T): We "soften" the probability distribution to exaggerate these small details so the student can learn them.

The Distillation Loss Function

- The Formula:
  - $L_{total} = \alpha L_{CE}(y, \hat{y}_{student}) + (1-\alpha) L_{KL}(P_{teacher}, P_{student})$
- Breakdown:
  - Student vs. Reality (L_{CE}): "Get the right answer."
  - Student vs. Teacher (L_{KL}): "Reason like the expert."
- White-box vs. Black-box:
  - White-box: We have access to the Teacher's weights/logits (Classic KD).
  - Black-box (Modern API Era): We only get the text output. We use "Synthetic Data Generation" (Teacher generates Q&A pairs, Student trains on them).

When to use Distillation?

- **High Volume / Low Latency**: Spam filters, Search ranking, On-device assistants.
- **Privacy**: Distill a cloud model into a local model that never sends data out.
- A successful example:



DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter

Victor SANH, Lysandre DEBUT, Julien CHAUMOND, Thomas WOLF
Hugging Face
{victor,lysandre,julien,thomas}@huggingface.co

Abstract

As Transfer Learning from large-scale pre-trained models becomes more prevalent in Natural Language Processing (NLP), operating these large models in on-the-edge and/or under constrained computational training or inference budgets remains challenging. In this work, we propose a method to pre-train a smaller general-purpose language representation model, called DistilBERT, which can then be fine-tuned with good performances on a wide range of tasks like its larger counterparts. While most prior work investigated the use of distillation for building task-specific models, we leverage knowledge distillation during the pre-training phase and show that it is possible to reduce the size of a BERT model by 40%, while retaining 97% of its language understanding capabilities and being 60% faster. To leverage the inductive biases learned by larger models during pre-training, we introduce a triple loss combining language modeling, distillation and cosine-distance losses. Our smaller, faster and lighter model is cheaper to pre-train and we demonstrate its capabilities for on-device computations in a proof-of-concept experiment and a comparative on-device study.

Conclusion

- Pre-training
- Pre-processing
- Evaluation of pre-training
- Fine-tuning of pre-trained models for supervised tasks
- Knowledge distillation