Advanced Deep Learning

Section 3:

Natural Language Processing

Rényi Alfréd Matematikai Kutatóintézet

ELTE EÖTVÖS LORÁND TUDOMÁNYEGYETEM

Schedule

Lectures on NLP:

- **Lecture 1: Introduction and Foundations**
  - Characteristics of the domain
  - Classical methods
  - Character encodings
  - Tokenization
  - Embeddings
- **Lecture 2: Language Models and Language Modeling**
  - Objective functions
  - Sequential modeling
  - Decoding strategies
  - Models: Transformers (BERT, GPT)
  - Training: pre-training, fine-tuning
  - Evaluation
- **Lecture 3: Large Language Models (LLMs)**
  - Emergent properties
  - Scaling laws
  - GPT-series
  - Instruction tuning
  - Reinforcement Learning with Human Feedback (RLHF)
- **Lecture 4: Research**
  - Prompt engineering
  - Multimodality: CLIP
  - Problems: hallucination
  - Retrieval Augmented Generation (RAG)
  - Security

Large Language Models (LLMs)

Lecture 3

What is this lecture about?

This lecture tries to answer the following questions:

- What are Large Language Models (LLMs)?
- Why do we increase the size of the Language Models?
- What makes Large Language Models such successful?
    - What are Emergent Abilities?
- What makes a Next Word Prediction model to be able to function in an Application?

Resource

The lecture content has been developed mainly from the paper:

- [A Survey of Large Language Models]

Introduction: LLMs

Large Language Models (LLMs):

- Deep Learning models:
    - designed to understand, generate, and manipulate human language in textual form
- Model Architecture:
    - typically Transformer-based
- Model size:
    - ranging from hundreds of millions to billions of parameters
- Capabilities:
    - able to perform a wide range of language-based tasks without task-specific tuning
- Advantages:
    - Versatility: performing multiple tasks
    - Efficiency: generating language with minimal human intervention
    - Real-World Applications: Customer Service; Content Generation, …

Scaling Laws

Recent trends: Continued Scaling:

- effort to build even larger and more powerful models continue

Scaling Laws:

- a quantitative approach to characterizing the scaling effect
- scaling can largely improve the model capacity:
    - Model size: number of parameters
    - Dataset size: number of training examples
    - Total compute: amount of training compute (time, iteration)

Scaling Laws

KM scaling law:

$$
\begin{aligned}
L(N) &= \left(\frac{N_c}{N}\right)^{\alpha_N}, \quad \alpha_N \sim 0.076, N_c \sim 8.8 \times 10^{13} \\
L(D) &= \left(\frac{D_c}{D}\right)^{\alpha_D}, \quad \alpha_D \sim 0.095, D_c \sim 5.4 \times 10^{13} \\
L(C) &= \left(\frac{C_c}{C}\right)^{\alpha_C}, \quad \alpha_C \sim 0.050, C_c \sim 3.1 \times 10^{8}
\end{aligned}
$$

Where:

- *L(●)*: the cross entropy loss in nats
- *N*: model size
- *D*: dataset size
- *C*: amount of training compute

Scaling Laws

Chinchilla scaling law:

$$L(N, D) = E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}$$

$$E = 1.69, A = 406.4, B = 410.7, \alpha = 0.34, \beta = 0.28$$

Where:

- *L(●)*: the cross entropy loss in nats
- *N*: model size
- *D*: dataset size

Scaling: performance

More text → Lower cross-entropy

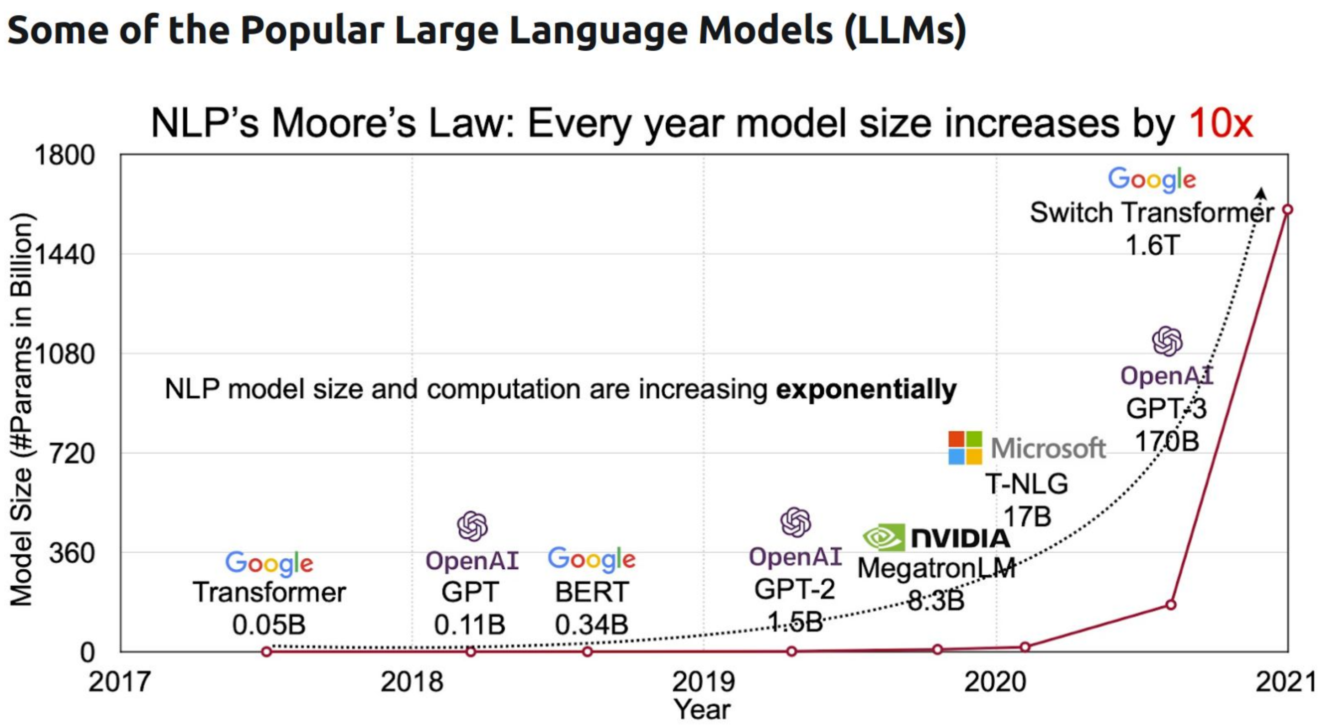
More parameters → Lower cross-entropy

Lower cross-entropy → Better performance

- correlating well with other evaluation metrics

Currently scaling does not show signs of topping out

Diagram of sizes



Model parameter sizes

Name	Release date[a]	Developer	Number of parameters (billion) [b]	Corpus size	Training cost (petaFLOP-day)	License[c]	Notes
GPT-1	June 2018	OpenAI	0.117		1[121]	MIT[122]	First GPT model, decoder-only transformer. Trained for 30 days on 8 P600 GPUs.
BERT	October 2018	Google	0.340[123]	3.3 billion words[123]	9[124]	Apache 2.0[125]	An early and influential language model,[7] but encoder-only and thus not built to be prompted or generative[126]
GPT-2	February 2019	OpenAI	1.5[134]	40GB[135] (~10 billion tokens)[136]		MIT[137]	general-purpose model based on transformer architecture
XLNet	June 2019	Google	~0.340[130]	33 billion words		Apache 2.0[131]	An alternative to BERT; designed as encoder-only[132][133]
GPT-3	May 2020	OpenAI	175[39]	300 billion tokens[136]	3640[138]	proprietary	A fine-tuned variant of GPT-3, termed GPT-3.5, was made available to the public through a web interface called ChatGPT in 2022.[139]

Emergent Abilities

Emergent Abilities:

- Qualitative changes in the model's behaviour
  - not trained explicitly
- **Emergent**: an ability is emergent if it is not present in smaller models but is present in larger models
  - arise from the quantitative increase in the scale of the Language Model
- the performance rises significantly when above random when the scale reaches a certain level
- emergent abilities cannot be predicted simply by extrapolating the performance of smaller models
  - the abilities appear suddenly and unpredictable, as the model reaches a critical threshold of scale, rather than improving steadily and predictably
- Examples:
  - In-Context Learning
  - Instruction following
  - Chain-of-Thought Prompting

Emergent Abilities: In-Context Learning

In-Context Learning (ICL):

- the ability of a Language Model to learn from the context provided in the input itself, without additional training or external data
    - a natural language instruction and / or
    - several task demonstrations
- introduced by GPT-3
- relying on the examples included in the input prompt to guide its responses

Zero-, One-, Few-shot Learning:

- training the model (updating the parameters) with one / few examples

Demo:

- ChatGPT: eszperente, arithmetic tasks

Emergent Abilities: Instruction following

Instruction following:

- fine-tuning the Language Model with datasets formatted via natural language descriptions (instruction tuning)
- LLMs perform well on unseen tasks that are also described in the form of instructions

Emergent Abilities: Chain-of-Thought Prompting

Chain of Thought (CoT) Prompting:

- a prompt engineering technique
- aims to improve language models’ performance on tasks requiring logic, calculation and decision-making
- by structuring the input prompt in a way that mimics human reasoning
- step-by-step reasoning
- Assumption: obtained by training on code

Training phases

Training phases / approaches:

- Pre-Training
- Adaptation / Fine-Tuning:
  - Instruction Tuning
  - Alignment Tuning
    - Reinforcement Learning with Human Feedback (RLHF)
- Prompting

Pre-training

Pre-training:

- optimizing for completion
  - ML task: Language Modeling
- training data:
  - large quantity
  - low quality
  - Scale: 1 trillion tokens (~ 15 million books)
- Examples:
  - GPT-x, LLaMa

Adaptation

Adaptation of LLMs:

- after pre-training
  - LLMs acquired the general abilities
- adapting LLM’s abilities to specific goals
- 2 kinds of adaptations:
  - Instruction Tuning
  - Alignment Tuning

Adaptation: Instruction Tuning

Instruction Tuning:

- fine-tuning pre-trained LLMs on a collection of formatted instances in the form of natural language
- Supervised Fine-Tuning
- Dataset:
    - instruction-formatted data
- Supervised training:
    - sequence-to-sequence loss
    - Input:
        - task description (instruction)
        - additional demonstrations (optional)
    - Output:
        - required answer / reaction
- abilities to generalized to unseen tasks
- Examples:
    - Existing NLP tasks: summarization, translation, text classification, question answering
        - augmenting them with human labeled task descriptions
    - Dialogue system / Open Question Answering
        - human labelers compose instructions for tasks and another group of labelers answer them

Adaptation: Alignment Tuning

Alignment Tuning:

- LLMs sometimes exhibit unintended behavior
    - false information
    - inaccurate objectives
    - harmful, misleading, biased outputs
- Pre-training objective:
    - lack of human values or preferences
- Alignment Tuning: making LLMs act in line with human expectations
    - Criteria:
        - Helpfulness: attempt to solve the given task in a concise and efficient manner
        - Honesty: accurate content instead of fabricated information (hallucination)
        - Harmlessness: not offensive or discriminatory

Adaptation: Alignment Tuning: Human Feedback

Alignment Tuning:

- **Collecting Human Feedback:**
  - high quality human feedback
  - Human Labeler Selection:
    - human labelers should have a qualified level of education and excellent proficiency
  - Human Feedback Collection:
    - Ranking-based:
      - evaluating model-generated outputs by raking them
        - easier to compare outputs than generate the best
      - not selecting the best answer(s)
      - more consistent answer (than selecting the best)
    - Question-based:
      - human labelers answer certain questions designed by researchers
      - covering the alignment criteria and additional constraints
      - task-specific questions based on the generated output
    - Rule-based:
      - response preference feedback: comparing the quality of output pairs
      - rule violation feedback: scoring the extent of the violation of the rules

Adaptation: Alignment Tuning: RLHF

Alignment Tuning:

- **Reinforcement Learning with Human Feedback (RLHF):**
    - OpenAI's approach
    - fine-tuning LLMs with the collected human feedback using Reinforcement Learning
    - Components:
        - Pre-trained Language Model
        - Reward model:
            - learning from the human feedback
            - provides the training signal for the Language Model
                - reflecting the human preferences
        - Reinforcement Learning algorithm:
            - Proximal Policy Optimization
            - training the Language Model

Adaptation: Alignment Tuning: RLHF

Alignment Tuning:

- **Reinforcement Learning with Human Feedback (RLHF):**
    - Steps of RLHF:
        - Supervised Fine-Tuning:
            - collecting a supervised dataset containing input prompts and desired outputs
            - fine-tuning the Language Model
            - prompts and outputs can be written by human labelers
        - Reward model training:
            - the Language Model generates a certain number of output texts
            - human labelers annotate the preference for these pairs
                - ranking the generated candidate texts
            - a Reward model is trained to predict the human-preferred output
        - RL Fine-Tuning:
            - aligning: fine-tuning the Language Model by a Reinforcement Learning algorithm
            - the pre-trained Language Model acts as the policy:
                - taking as input a prompt
                - returning an output text
                    - action space: the vocabulary
                    - state: currently generated token sequence
                    - reward: provided by the Reward model
        - Regularization:
            - to avoid diverging significantly from the initial Language Model
            - a penalty term: KL-divergence
                - between the generated results from the current Language Model and the initial Language Model

Adaptation: Alignment Tuning: RLHF

Advantages of RLHF:

- helping with hallucination
- Why RLHF works?
  - Diversity:
    - Supervised Fine-Tuning:
      - the LM’s output is expected to match the demonstrated responses
        - demonstration only gives the model positive signals
          - no negative signals
    - RLHF:
      - the model gets negative feedback as well
      - RL allows to show negative signals as well

Prompting

Prompting:

- no parameter update to the LLM
- using In-Context Learning abilities:
    - Instructions:
        - specifying what to do
    - Descriptions:
        - detailing and explaining the task
    - Examples:
        - providing example solutions

Comparisons

Pre-training vs. Fine-tuning

Fine-tuning vs. Prompting

Pre-Training vs. Supervised Fine-Tuning

Pre-training stage:

- is about knowledge
- Data:
  - raw text
  - low quality
  - large quantity

Supervised fine-tuning stage:

- is about alignment
- learning to respond more consistently
- increasing knowledge of new specific concepts
- correcting old incorrect information
- Data:
  - demonstration data: in the proper format (prompt, response)
  - high quality data
  - smaller quantity
    - 10,000 - 100,000 pairs
  - format: (prompt, response)
- fine-tuning stage is much cheaper:
- companies do it a lot more frequently than pre-training

Pre-Training vs. Supervised Fine-Tuning

Pre-training stage:

- is about knowledge
- Data:
  - raw text
  - low quality
  - large quantity

Supervised fine-tuning stage:

- is about alignment
- learning to respond more consistently
- increasing knowledge of new specific concepts
- correcting old incorrect information
- Data:
  - demonstration data: in the proper format (prompt, response)
  - high quality data
  - smaller quantity
    - 10,000 - 100,000 pairs
  - format: (prompt, response)
- fine-tuning stage is much cheaper:
- companies do it a lot more frequently than pre-training

Fine-tuning vs. Prompting

Prompting:

- rather go by prompting first then fine-tuning
- Pros:
    - no data to get started
    - smaller upfront cost
    - no technical knowledge needed
    - connect data through retrieval (RAG)
    - lower cost during training
- Cons:
    - much less data fits
    - forgets data
- great for generic, side projects, prototypes

Fine-tuning:

- customizing the model to a specific use case
- easier to show than tell
- typical clear improvements from 50-100 examples
- fine-tuning lets us put more data into the model than what fits into the prompt
- Pros:
    - nearly unlimited data fits
    - not forgetting data
    - reducing hallucination
    - lower cost during inference
- Cons:
    - more high-quality data
    - upfront compute cost
    - needs some technical knowledge, esp. data
- great for domain-specific, enterprise, production usages, … privacy!

Fine-tuning vs. Prompting

Prompting:

- rather go by prompting first then fine-tuning
- Pros:
  - no data to get started
  - smaller upfront cost
  - no technical knowledge needed
  - connect data through retrieval (RAG)
  - lower cost during training
- Cons:
  - much less data fits
  - forgets data
- great for generic, side projects, prototypes

Fine-tuning:

- customizing the model to a specific use case
- easier to show than tell
- typical clear improvements from 50-100 examples
- fine-tuning lets us put more data into the model than what fits into the prompt
- Pros:
  - nearly unlimited data fits
  - not forgetting data
  - reducing hallucination
  - lower cost during inference
- Cons:
  - more high-quality data
  - upfront compute cost
  - needs some technical knowledge, esp. data
- great for domain-specific, enterprise, production usages, … privacy!

GPT series

Technical evolution of the GPT-series models:

- 2 key points to the success:
  - training decoder-only Transformer LMs that can accurately predict the next word
  - scaling up the size of LMs
- Initial GPT models:
  - **GPT-1**
  - **GPT-2**
- Large Language Models (LLMs):
  - **GPT-3**
  - **Fine-tuned models**
  - **GPT-3.5**
  - **ChatGPT**
  - **GPT-4**

GPT series

Technical evolution of the GPT-series models:

- 2 key points to the success:
  - training decoder-only Transformer LMs that can accurately predict the next word
  - scaling up the size of LMs
- Initial GPT models:
  - **GPT-1**
  - **GPT-2**
- Large Language Models (LLMs):
  - **GPT-3**
  - **Fine-tuned models**
  - **GPT-3.5**
  - **ChatGPT**
  - **GPT-4**

- in 2018
- the Transformer architecture is adopted (change RNNs)
- unsupervised pre-training and supervised fine-tuning
- 117 million parameters

GPT series

Technical evolution of the GPT-series models:

- 2 key points to the success:
  - training decoder-only Transformer LMs that can accurately predict the next word
  - scaling up the size of LMs
- Initial GPT models:
  - **GPT-1**
  - **GPT-2**
- Large Language Models (LLMs):
  - **GPT-3**
  - **Fine-tuned models**
  - **GPT-3.5**
  - **ChatGPT**
  - **GPT-4**

- "Unsupervised Multitask Learner"
- in 2019
- 1.5 billion parameters
- unsupervised language modeling (no explicit fine-tuning on labeled data)
  - p(output | input, task)
    - conditioned on the input and the task information
    - NLP tasks can be considered as the word prediction problem based on a subset of the world text
      - unsupervised LMs could be capable in solving various tasks
    - GPT-2 paper: "Language Models are Unsupervised Multitask Learners"
    - inferior performance compared with supervised fine-tuning SOTA methods

GPT series

Technical evolution of the GPT-series models:

- 2 key points to the success:
  - training decoder-only Transformer LMs that can accurately predict the next word
  - scaling up the size of LMs
- Initial GPT models:
  - **GPT-1**
  - **GPT-2**
- Large Language Models (LLMs):
  - **GPT-3**
  - **Fine-tuned models**
  - **GPT-3.5**
  - **ChatGPT**
  - **GPT-4**

- in 2020
- 175 billion parameters
- introducing: In-Context Learning (ICL)
  - utilizing LLMs in a few-shot or zero-shot manner
  - ICL can instruct a LLMs to understand the tasks in the form of natural language text
  - pre-training and utilization of LLMs converge to the same language modeling paradigm
- excellent performance in a variety of NL tasks
  - large performance leap
  - scaling law: larger models have significantly stronger ICL abilities

GPT series

Technical evolution of the GPT-series models:

- 2 key points to the success:
  - training decoder-only Transformer LMs that can accurately predict the next word
  - scaling up the size of LMs
- Initial GPT models:
  - **GPT-1**
  - **GPT-2**
- Large Language Models (LLMs):
  - **GPT-3**
  - **Fine-tuned models**
  - **GPT-3.5**
  - **ChatGPT**
  - **GPT-4**

- 2 major approaches to further improve the GPT-3 model
- Training on code data: **Codex**
  - GPT-3: lacks reasoning ability on complex tasks
  - in 2021
  - a GPT model fine-tuned on a large corpus of GitHub code
  - solving complex coding tasks and math problems
  - improving reasoning abilities and chain-of-thought prompting abilities
- Alignment with human preference: **InstructGPT**
  - in 2022
  - improving GPT-3 for human alignment
    - issues of generating harm or toxic content
    - key to the safe deployment of LLMs in practice
  - 3-stage Reinforcement Learning from Human Feedback (RLHF)

GPT series

Technical evolution of the GPT-series models:

- 2 key points to the success:
  - training decoder-only Transformer LMs that can accurately predict the next word
  - scaling up the size of LMs
- Initial GPT models:
  - **GPT-1**
  - **GPT-2**
- Large Language Models (LLMs):
  - **GPT-3**
  - **Fine-tuned models**
  - **GPT-3.5**
  - **ChatGPT**
  - **GPT-4**

- based on a code-based and RLHF-based GPT models
- in 2022

GPT series

Technical evolution of the GPT-series models:

- 2 key points to the success:
  - training decoder-only Transformer LMs that can accurately predict the next word
  - scaling up the size of LMs
- Initial GPT models:
  - **GPT-1**
  - **GPT-2**
- Large Language Models (LLMs):
  - **GPT-3**
  - **Fine-tuned models**
  - **GPT-3.5**
  - **ChatGPT**
  - **GPT-4**

- conversation model
- in 2022
- based on GPT-3.5 (and later GPT-4)
- trained in a similar way as InstuctGPT
  - specially optimized for dialogue
- plugin mechanism:
  - further extending the capabilities of ChatGPT with existing tools or apps

GPT series

Technical evolution of the GPT-series models:

- 2 key points to the success:
  - training decoder-only Transformer LMs that can accurately predict the next word
  - scaling up the size of LMs
- Initial GPT models:
  - **GPT-1**
  - **GPT-2**
- Large Language Models (LLMs):
  - **GPT-3**
  - **Fine-tuned models**
  - **GPT-3.5**
  - **ChatGPT**
  - **GPT-4**

- in 2023
- extending the text input to multimodal signals
- better performance on evaluation tasks than GPT-3.5
- Red Teaming:
  - Goal: reducing the harm or toxic content generation
  - using manual or automated methods to adversarially probe a language model for harmful outputs, and then updating the model to avoid such outputs
- Predictable scaling:
  - accurately predicting the final performance with a small proportion of compute during model training

Proprietary vs. Open-source Models

Comparison between closed and open models:

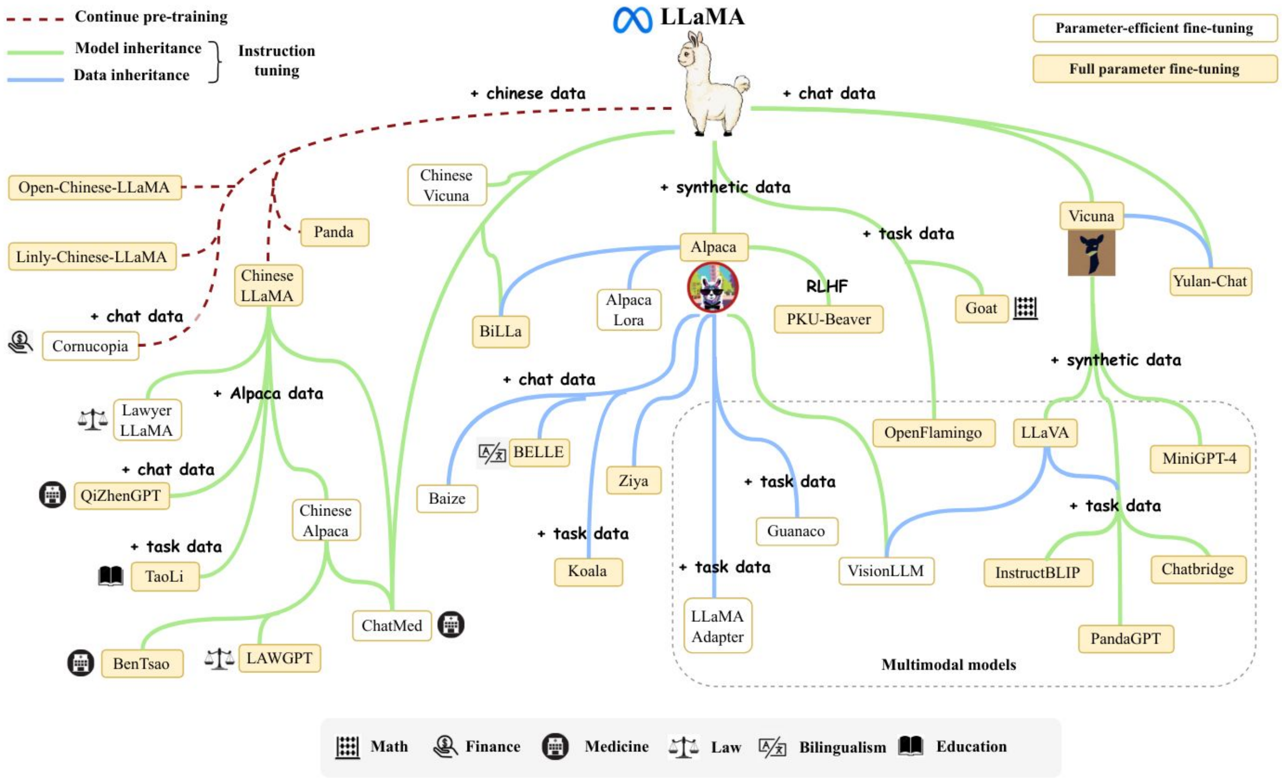
Closed models:

- better performance
- we cannot
  - really work with them
  - fine-tune with them
  - download them

Open-source models:

- lower performance
- depending on our application can be good enough
- we can fine-tune

LLaMa model series



Source

LLaMa model series

LLaMa:

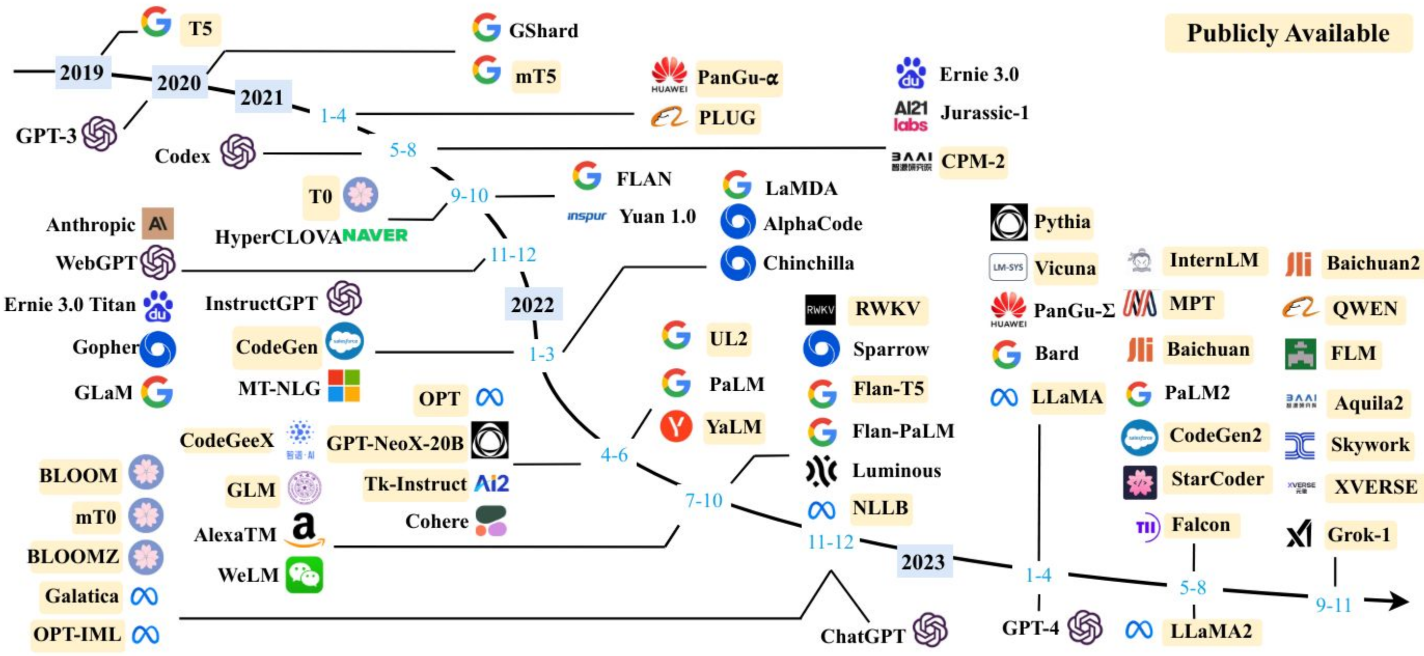
- probably the most powerful open weights model
- weights, architecture, paper: all released by Meta
- example model: llama-2-70b:
- 70 billion parameters:
- a parameters: float 16 → 140 GB of size
- releasing both base model and fine-tuned model
- training:
- data: chunk of the internet: 10TB of text
- 6000 GPUs for 12 days
- 140 GB model size
- 2 million dollars cost

Hungarian LLMs

Magyar NLP modellek:

- https://acta.bibl.u-szeged.hu/78417/1/msznykonf_019_247-262..pdf
- Hugging Face - NYTK modellek (Nyelvtudományi Kutatóközpont):
    - https://huggingface.co/NYTK
        - Modellek:
            - PULI-GPTrio
            - named-entity-recognition-nerkor-hubert-hungarian
            - PULI-GPT-3SX
            - sentiment-hts5-xlm-roberta-hungarian
            - sentence-transformers-experimental-hubert-hungarian
            - PULI-GPT-2
            - PULI-BERT-Large
        - Adathalmazok:
            - HuCOLA
            - HuSST
            - HuCoPA
            - HuWNLI
            - HuRC

List of LLMs



[Source](#)

List of LLMs

(Large) Language Models:

- BERT: https://arxiv.org/abs/1810.04805
- GPT-1: https://openai.com/research/language-unsupervised
- GPT-2: https://openai.com/research/better-language-models
- GPT-3: https://openai.com/research/language-models-are-few-shot-learners
- GPT-4: https://openai.com/research/gpt-4
- ChatGPT: https://openai.com/blog/chatgpt
- XLNet: https://arxiv.org/abs/1906.08237
- RoBERTa: https://arxiv.org/abs/1907.11692
- ALBERT: https://arxiv.org/abs/1909.11942
- T5: https://arxiv.org/abs/1910.10683v4
- ELECTRA: https://arxiv.org/abs/2003.10555
- DeBERTa: https://arxiv.org/abs/2006.03654
- PaLM: https://arxiv.org/abs/2204.02311
- ELMO: https://arxiv.org/abs/1802.05365
- ULMFiT: https://arxiv.org/abs/1801.06146
- DistilBERT: https://arxiv.org/abs/1910.01108
- XLM-RoBERTa: https://arxiv.org/abs/1911.02116
- UniLM: https://arxiv.org/abs/1905.03197
- StructBERT: https://arxiv.org/abs/1908.04577
- MobileBERT: https://arxiv.org/abs/2004.02984
- CTRL: https://arxiv.org/abs/1909.05858
- Flair: https://aclanthology.org/N19-4010.pdf
- Llama: https://arxiv.org/abs/2302.13971
- Llama-2: https://arxiv.org/abs/2307.09288
- Transformer-XL: https://arxiv.org/abs/1901.02860
- ERNIE: https://arxiv.org/abs/1905.07129
- BLIP: https://arxiv.org/abs/2201.12086
- BLIP-2: https://arxiv.org/abs/2301.12597
- Flamingo: https://arxiv.org/abs/2204.14198
- OpenFlamingo: https://arxiv.org/abs/2308.01390
- LLaVA: https://llava-vl.github.io/

ChatGPT API

https://openai.com/index/introducing-chatgpt-and-whisper-apis

https://help.openai.com/en/collections/3675931-api

Additional resources

[A Survey of Large Language Models]