Pattern Recognition
and Applications Lab

Introduction to LLMs

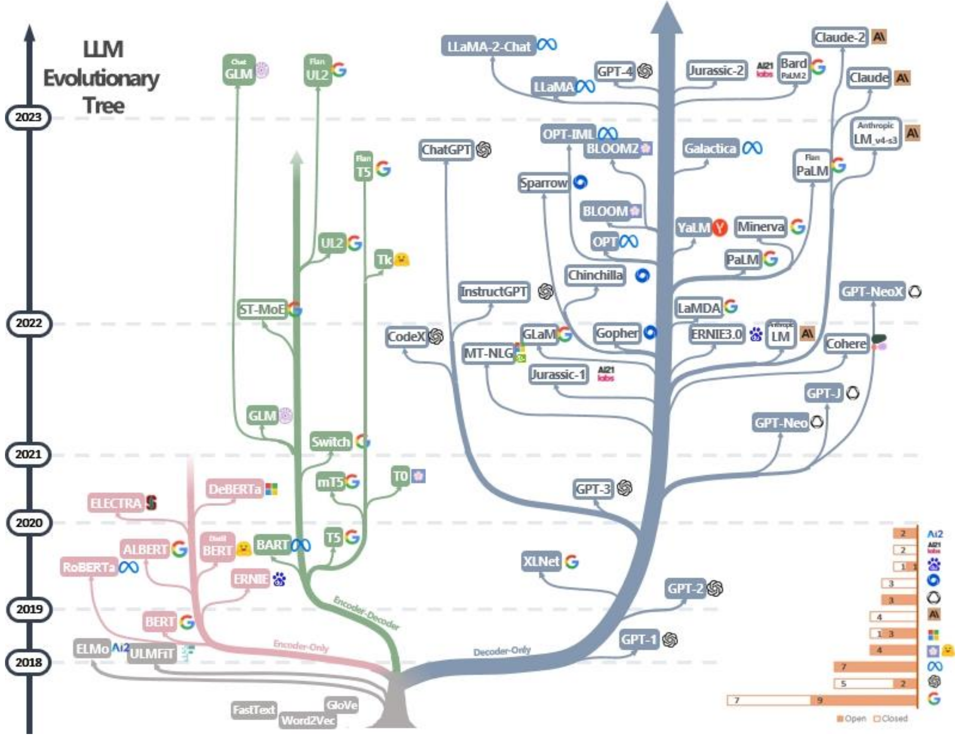
Angelo Sotgiu
angelo.sotgiu@unica.it

The new era of Deep Learning...

- Large Language Models are the most hyped ML technology in the last years

- Able to capture context, structure, and relationships within text

- Impressive performance on a wide range of task
  - translation
  - text generation and summarization
  - question answering
  - code completion and generation



Yang et al. "*Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond.*"

Ingredients of LLMs success

- **Transformer** architecture

- Huge training datasets
  - GPT1: ~4.5 GB
  - GPT2: ~40 GB
  - GPT3: ~570 GB
  - GPT4: unknown, but also images are included

- Self-supervised **pre-training** + supervised (or with human feedback) **fine-tuning**

- Models with billions of parameters
  - GPT1: 117 million parameters
  - GPT2: 1.5 billion parameters
  - GPT3: 175 billion parameters
  - GPT4: (rumors) ~1,78 trillions parameters

Basic Concepts

- Tokenization
- Embedding
- Attention
- Transformer Architecture
- Self-supervised pre-training
- Fine-tuning

From text to tokens

- A token is a unit of text, composed of one or more characters (including symbols)

- Simple tokenization:

[A] [token] [is] [a] [unit] [of] [text] [,] [composed] [of] [one] [or] [more] [characters] [(] [including] [symbols] [)]

- A vocabulary is created from training data, mapping tokens to numerical values (IDs)

{ A: 0, token: 1, is: 2, a: 3, unit: 4, of: 5, …. }

- Special tokens are used as well (beginning/end of sequence, padding, unknown...)

- Problem: with very large datasets, the vocabulary size might explode

Tokenization algorithm

- Some algorithms allow to reduce the vocabulary size with subword tokenization, e.g., Wordpiece and Byte Pair Encoding (BPE)
- They basically detect (with statistical approaches) the most common subwords and splits the words accordingly
- For instance, common "-s" and "-ing" word endings can be considered as separate tokens
- In this way, the vocabulary will store a single token for both singular and plural words, and will not require additional tokens for "-ing" verbal forms

Text embedding

- A text embedding is a mapping from token IDs to a complex representation space, that can reflect complex relationships between tokens (for instance, similar words would lie close in the embedding space)
- It can be obtained by training specific layers or more complex neural networks
- The embedding space can have different dimensionality (from 1 to thousands)
- LLMs have their own embedding layer, trained together with them: in this way, the obtained representation is optimized based on their specific requirements
- After the training phase, the embedding layer is simply a lookup table mapping each token ID to its embedding vector

From RNNs to attention

- Text is a **sequence**, but classic sequence models have difficult to capture the relationships between words and the context

- For instance, older Recurrent Neural Networks (RNNs) encoded all the text sequence in a single hidden state, that was then processed by a decoder
  - issues with long sentences and long-term dependencies

- To address this issue, in 2014 an **attention** mechanism was added to RNNs
  - makes the decoder able to selectively access different parts of the input sequence at each decoding step

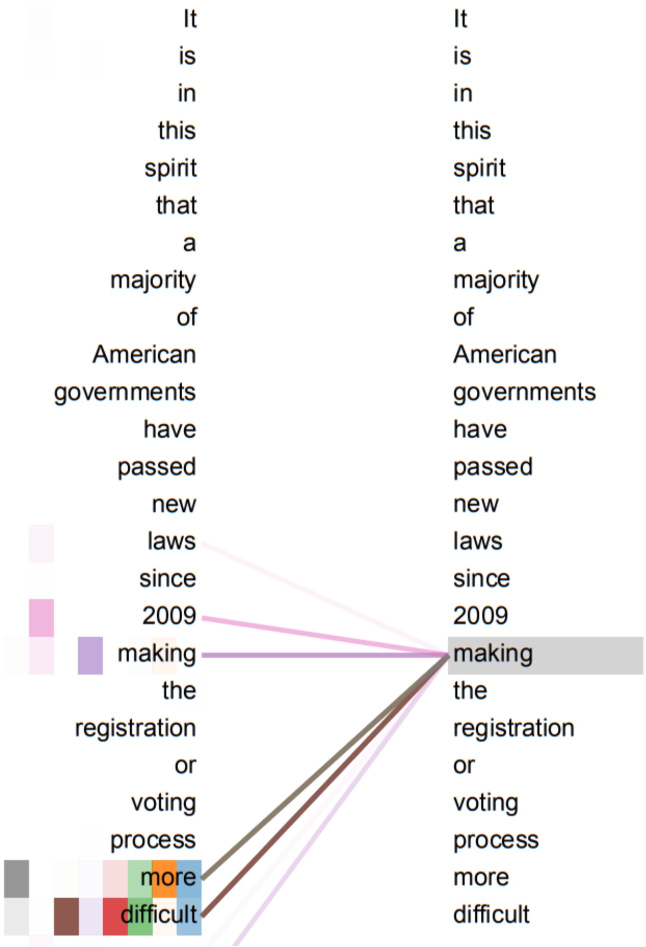
- Then, in 2017 it has been shown that by only relying on the attention mechanism (without RNNs) it was possible to build a very efficient model

Attention

- Attention is a mechanism that, given two sequences, allows the model to focus on the elements that determine the most important relationships between them
  - attention weights are computed

- **Self-attention** is computed on a single sequence, and allows the model to focus on the most important elements of the input when producing the output

- For each input token (actually, its embedding vector), a context vector is computed based on its relationships with all input tokens
  - to obtain meaningful representations, trainable parameters are involved in this operation



Waswani, A., et al. "*Attention is all you need.*" *NeurIPS 2017.*

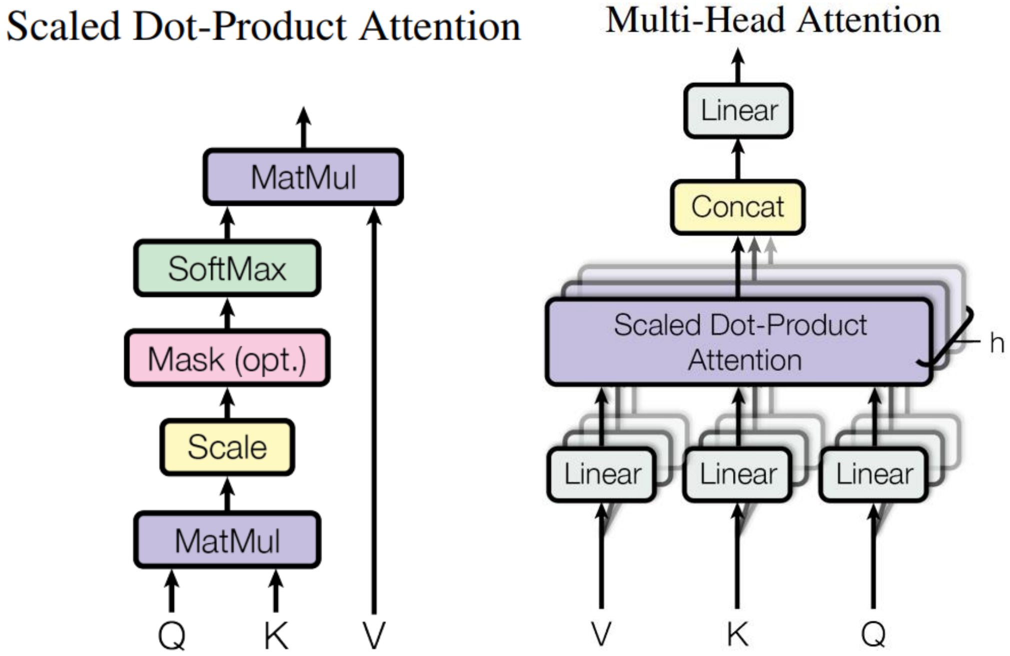
Attention

- Trainable parameter matrices: W_q, W_k, W_v

- Input embeddings are multiplied with them to obtain:
  - Query matrix
  - Key Matrix
  - Values Matrix

- Attention weights are computed by multiplying Q and K, scaling and normalizing

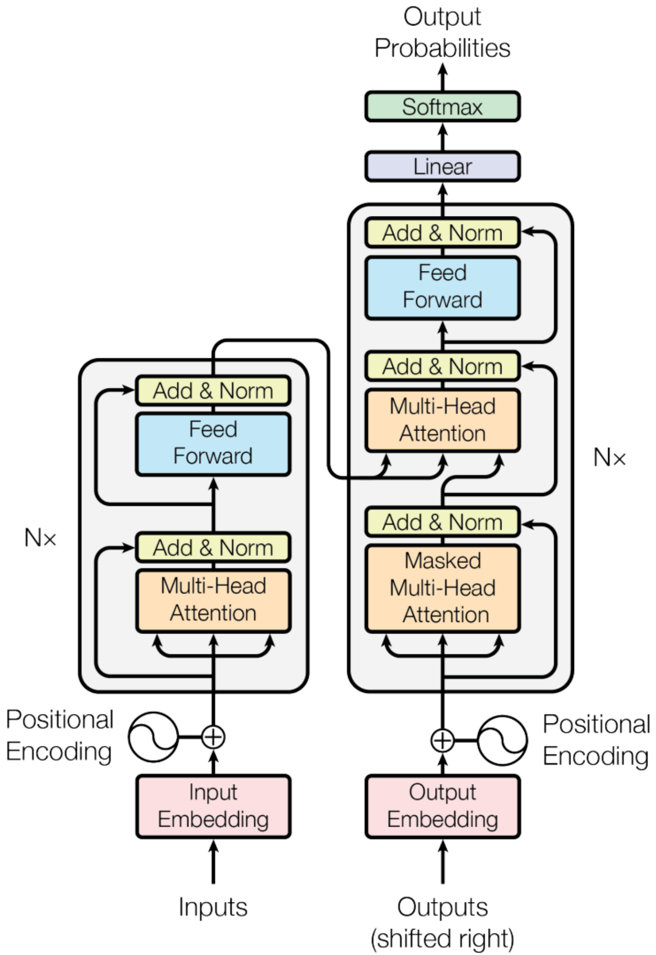
$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



Waswani, A., et al. "*Attention is all you need.*" *NeurIPS 2017.*

The Transformer

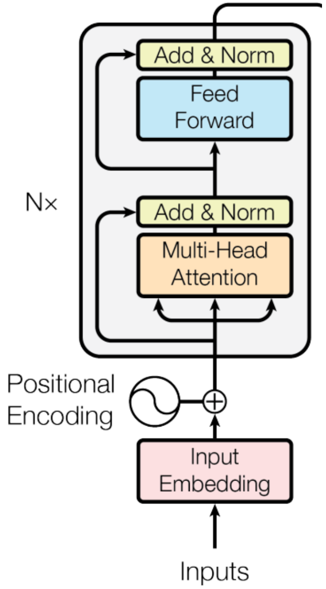
- Originally proposed as sequence-to-sequence model
- Composed of an Encoder (left side) and a Decoder (right side)
- Highly parallelizable and scalable
- Almost all LLMs are variants of this architeture



Waswani, A., et al. "*Attention is all you need.*" *NeurIPS 2017.*

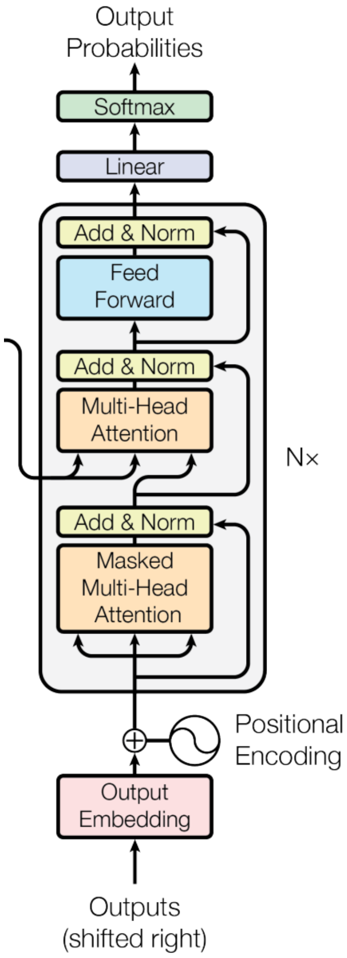
The Encoder

- Input text is tokenized, projected in the input embedding space and enriched with positional encoding
- At each stage, the attention layers can access all the tokens in the initial sequence
- The shortcuts help stabilize training
- The output is a mapping of the input text in the embedding space of the Encoder



The Decoder

- Takes as input the embedding vector of the input from the decoder, and the output at the previous stage (this makes it **autoregressive**)
- The output is a probability distribution over the token space, i.e., the predicted next token
- At each stage, for a given token the attention layers can only access the previous ones in the sequence
- The Decoder can be used alone

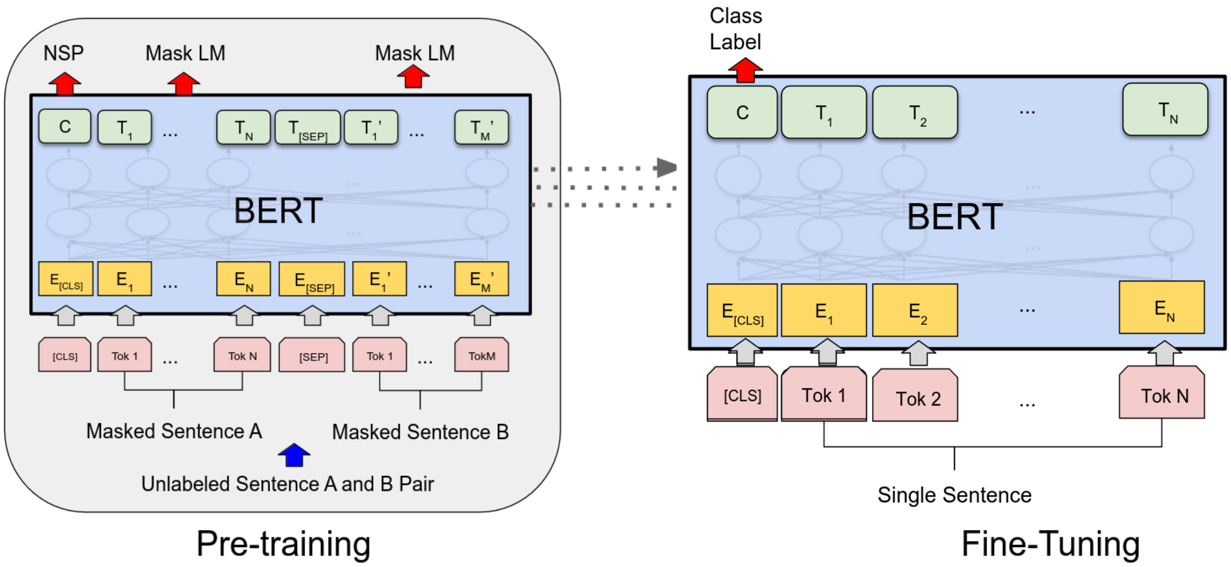


Encoder-only models

- Given a sequence of tokens, map it in a representation space
- Usually pre-trained with **masked language modeling**: random words are masked (removed) from sentences, and the model tries to fill them
- Additionally, it can be combined with **next sentence prediction** objective: given two input sentences, the model tries to predict whether the second follows the first
- Popular models: BERT, RoBERTa
- Particularly suited for text classification tasks

Encoder-only models

- The produced mapping can be used to train an additional layer on many downstream tasks



Devlin, J., et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* 2019

Decoder-only models

- Given a sequence of tokens, try to predict the next one
- Actually, they include both and encoder and a decoder
- Popular models: GPT, LLaMA, Claude, ….
- Used for text generation tasks

Encoder-Decoder models

- Given a sequence of tokens, generate another sequence of tokens

- Can be trained with several techniques, such as masking or deleting random tokens, masking multiple tokens with a single mask token, permuting sentences, rotating the document to make it start at a specific token, etc.

- Popular models: BART, T5

- Used for summarization and translation

Pre-training and N-shot learning

- Self-supervised pre-training is it not sufficient to reach the best performance
- Despite models are not pre-trained on any specific task, they might exhibit some ability
- Zero-shot: the model is asked to perform a new task without any example
- Few-shot (one-shot, two-shot, ...): the model is asked to perform a new task after providing some example

Fine tuning

- After pre-training, model are usually fine-tuned to perform specific tasks and to refine their output (for instance, adding security and safety guardrails)

- Fine-tuning on downstream tasks with labeled datasets

- Instruction-finetuning: a labeled dataset composed of instruction and answer pairs is used

- Reward-based learning with human feedback

Are LLMs generative models?

- Until now we have seen **discriminative** models
  - model the conditional probability distribution P(Y | X)
  - learn a function from training data that maps inputs to a target value
  - used for classification and regression

- Generative models aim to model the joint probability distribution P(X,Y)
  - learn a function to model the training data distribution and map
  - several proposed architectures:
    Generative Adversarial Networks (GANs), Diffusion Models, Variational Autoencoders

- LLMs are basically **token predictors**
  - formally they are not generative models
  - ...but they can *generate* data

References

- https://huggingface.co/learn/nlp-course
- Sebastian Raschka. Build A Large Language Model (From Scratch). 2023
- https://github.com/rasbt/LLMs-from-scratch