Natural Language Processing

Lecture 6 : Large Language Models

Master Degree in Computer Engineering
University of Padua
Lecturer : Giorgio Satta

Lecture partially based on material originally developed by :
Elena Voita, University of Edinburgh

Review: transformer

©Marvel Productions

Review: transformer

The **transformer** is a neural network architecture modeling sequence to sequence transformation (seq2seq): it turns one sequence into another sequence.

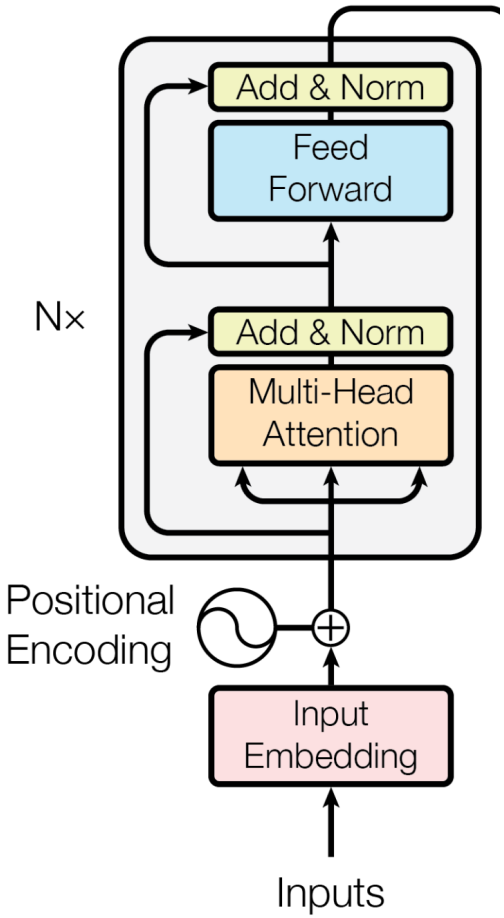
See Jurafsky & Martin §10.1 for definition of transformers.

The original transformer model uses an **encoder-decoder** architecture.

The transformer computes representations of its input and output entirely based on **self-attention**, without relying on sequential computation.

High degree of parallelism: computation performed for each item is independent of all the other computations.

Review: transformer



Encoder

- tokenizer + embedding layer with positional encoding
- stack of N identical blocks with two layers each
- first layer is a multi-head self-attention mechanism
- second layer is a fully connected feed-forward network
- both layers employ residual connection + normalization

Review: transformer

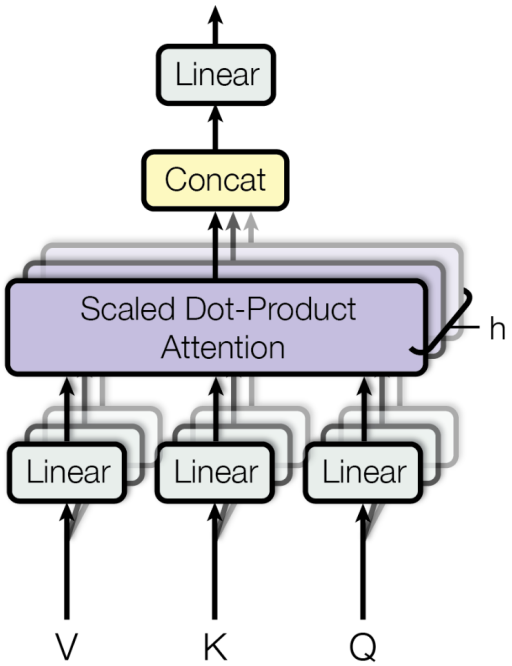
The encored uses so-called **scaled dot-product attention**.

Attention is computed on the basis of three different roles for the input tokens, produced through linear transformation

- **query**: current focus in the comparison
- **key**: importance to the given query
- **value**: contribution to the given query

Query-key product needs to be scaled for gradient stability.

Review: transformer



Multi-head attention

- input tokens can relate to each other in many different ways simultaneously
- multi-head attention implements parallel layers of self-attention that reside at the same depth

Review: transformer

Residual connection allows information to skip a layer, improving learning.

Layer normalization improves performance by keeping the values of a hidden layer in a range that facilitates gradient-based training.

This uses mean, standard deviation, gain and offset.

Feed forward: a simple MLP applied to each token individually; uses GeLU activation function.

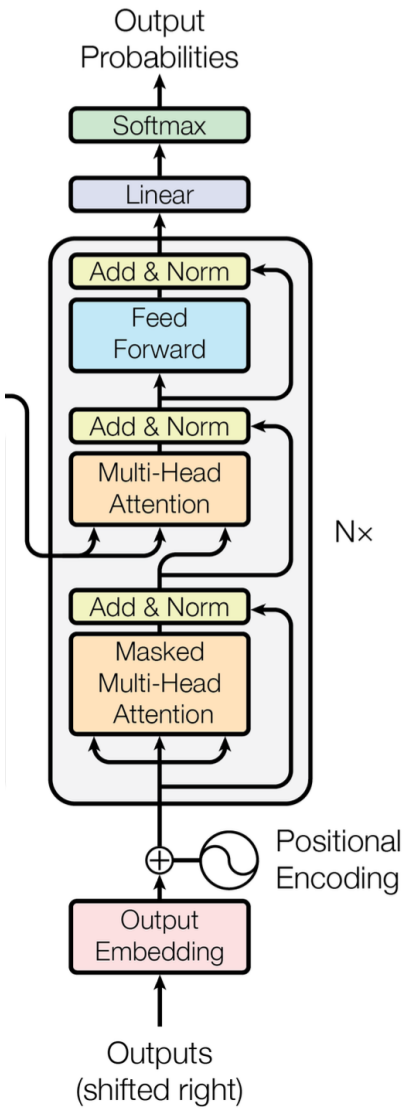
There's some weak evidence this is where "world knowledge" is stored.

Review: transformer

Input length is **fixed**, since attention is quadratic.

Encoder output is a high level, contextualized version of the input tokens, serving as a basis for decoder computation.

Review: transformer



Decoder

- stack of N identical blocks with three layers each
- first layer as in encoder, but with masked attention (auto-regressive)
- second layer uses cross-attention (key and value of encoder stack)
- third layers as in encoder
- all layers employ residual connection + normalization

Review: transformer

At training time, decoder uses **masked** self-attention. In this way, each token only sees the already generated ones and computation can be carried out in parallel.

Also called autoregressive decoding.

At generation time we need to generate one token at a time. This is why autoregressive decoding is extremely slow.

Review: transformer

Moving layer normalization before multi-headed attention and feedforward layers stabilizes training.

Fixed sinusoidal position embeddings sometimes replaced by unique, learned embeddings per position.

Some applications use encoder-only, or else decoder-only components.

Contextualized Embeddings

Magda Ehlers on Pexels

Contextualized Embeddings

Word embeddings such as word2vec or GloVe learn a single vector for each **type** (unique word) in the vocabulary V. These are also called **static embeddings**.

By contrast, **contextualized embeddings** represent each **token** (word occurrence) by a different vector, according to the context the token appears in.

Also called pre-trained language models, dynamic embeddings, large language models, or foundation model.

Incorporating context into word embeddings has proven to be a **watershed idea** in NLP, opening the way to transfer learning.

We discuss transfer learning and fine-tuning later on in this lecture.

Contextualized Embeddings

How can we learn to represent words along with the context they occur in?

We train a neural network combining ideas from word embeddings and language models:

- predict a word from left context

  GPT-*n* family
- predict a word from left and right context **independently**

  ELMo
- predict a word from left and right context **jointly**

  BERT

Presentation follows historical order. All presented models are language models.

©PBS/HBO Sesame Street

ELMo (embeddings from language model) looks at the entire sentence before assigning each word its embedding.

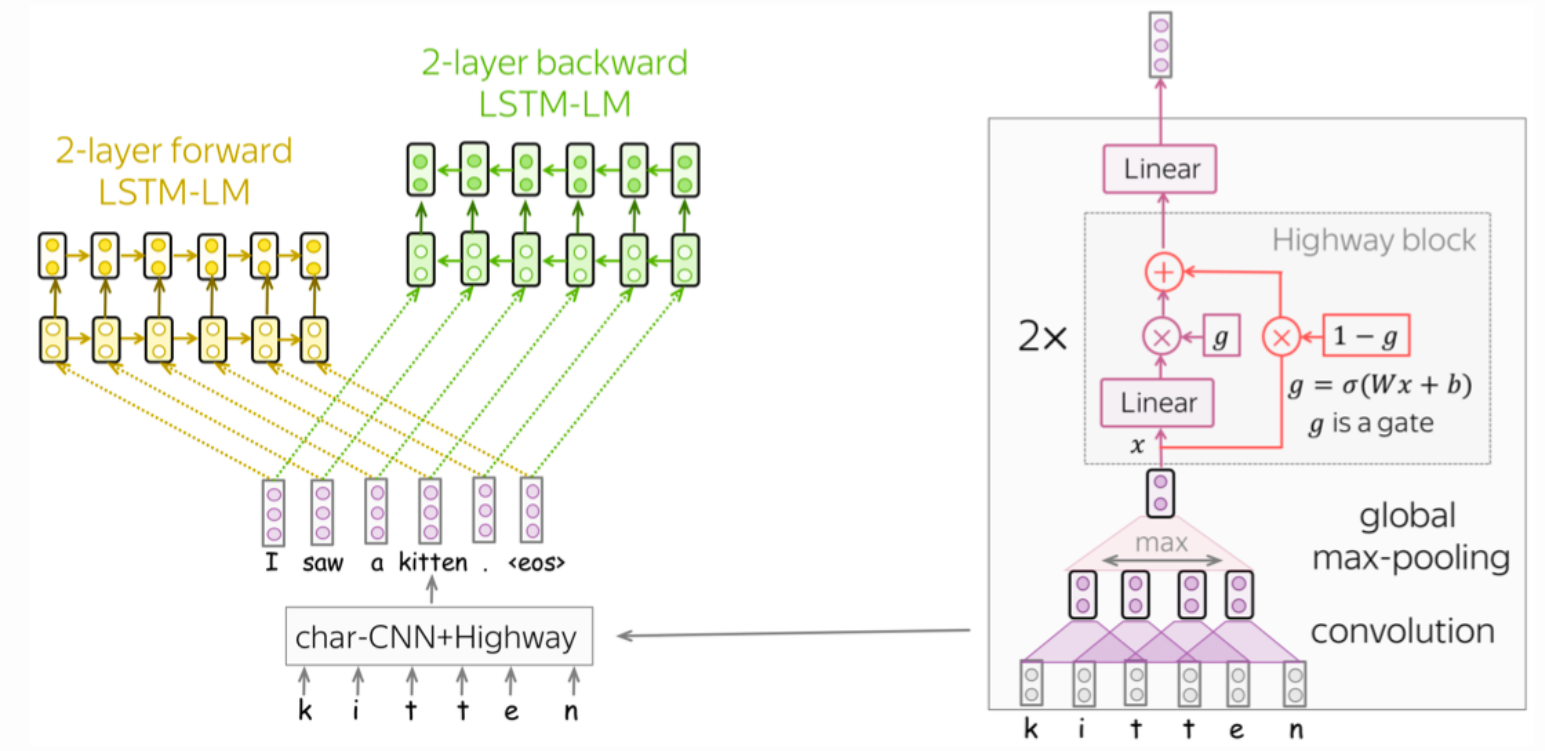
Created by researchers at the Allen Institute for Artificial Intelligence (AI2) and University of Washington, 2018.

Tokens are processed by a character-level convolutional neural network (CNN) producing word-level embeddings.

This captures word morphology and resolves OOV words.

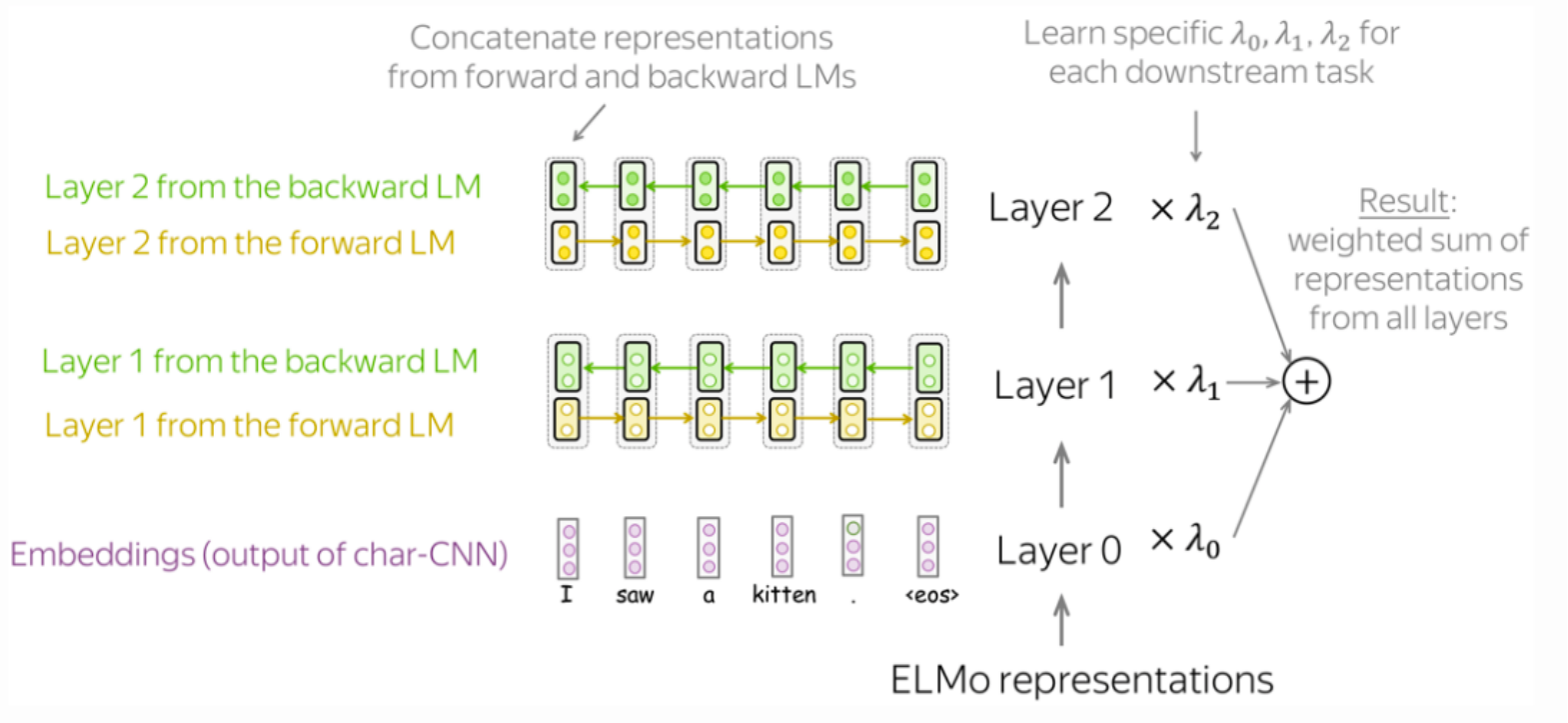
These embeddings are processed by a left-to-right and a right-to-left, 2-layers LSTM.

This is also called a bi-directional LSTM.



https://lena-voita.github.io/nlp_course/transfer_learning.html

For each word, the output embeddings at each layer (including the CNN layer) are combined, producing contextual embeddings.



https://lena-voita.github.io/nlp_course/transfer_learning.html

When **training**, add output layer with

- linear transformation to $|V|$ dimension
- softmax

and use cross-entropy as in neural language models.

Also recommended parameter dropout and L2 regularization.

When **encoding** the input

- ignore output layer
- retain the word embedding represented by the last hidden layer.

BERT

©PBS/HBO Sesame Street

BERT

BERT (bidirectional encoder representations from transformers) produces word representations by **jointly** conditioning on both left and right context.

Created by researchers at Google AI Language, 2018.

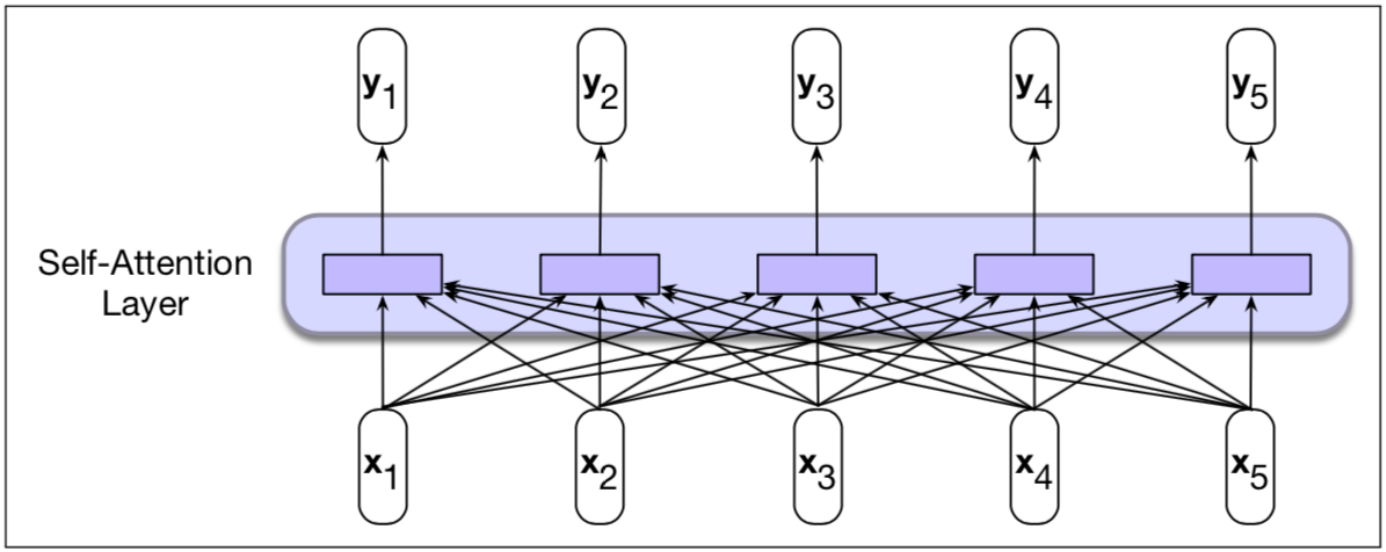
The model is based on the encoder component of the transformer neural network. Tokenizer uses WordPiece algorithm.

Two versions

- BERT Base: 12 layers, 12 attention heads, 768 embedding size, 110 million parameters
- BERT Large: 24 layers, 16 attention heads, 1024 embedding size, 340 million parameters

BERT

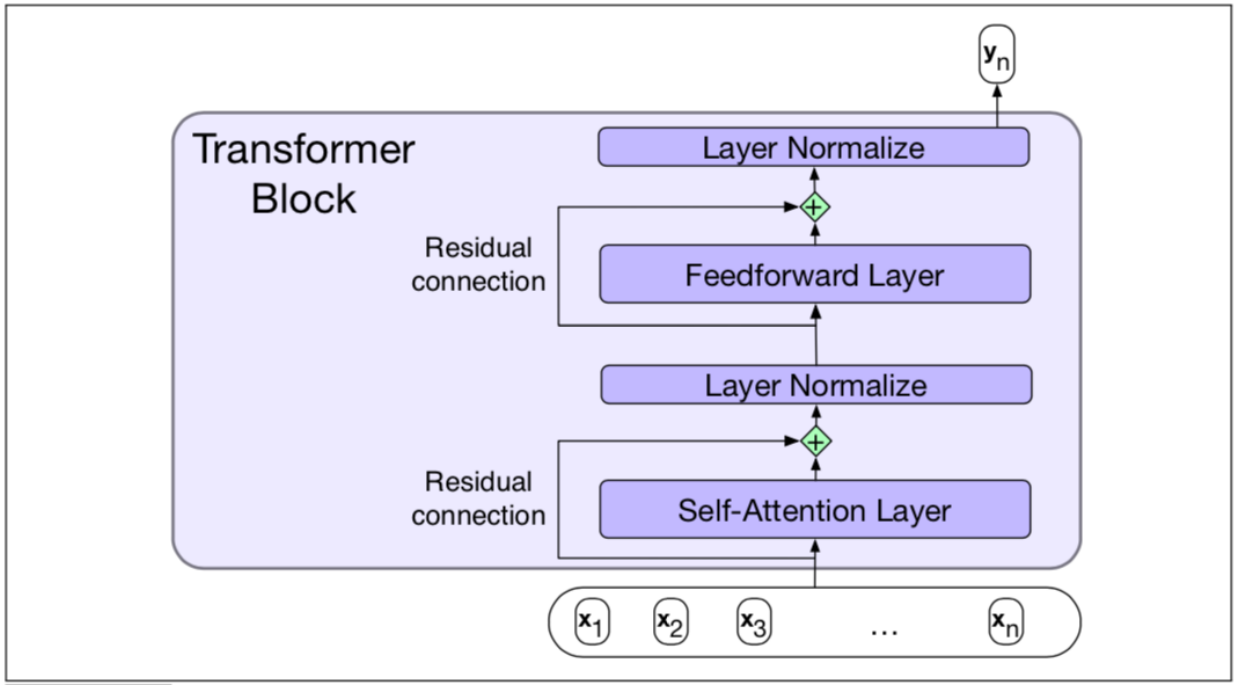
The bidirectional encoding in BERT is inherited from the self-attention mechanism of the transformer encoder.



BERT

Input is segmented using subword tokenization and combined with positional embeddings.

Then input is passed through a series of standard transformer blocks consisting of self-attention and feedforward layers, augmented with residual connections and layer normalization.



Masked language modeling

The model learns to perform a fill-in-the-blank task, technically called the **cloze task**.

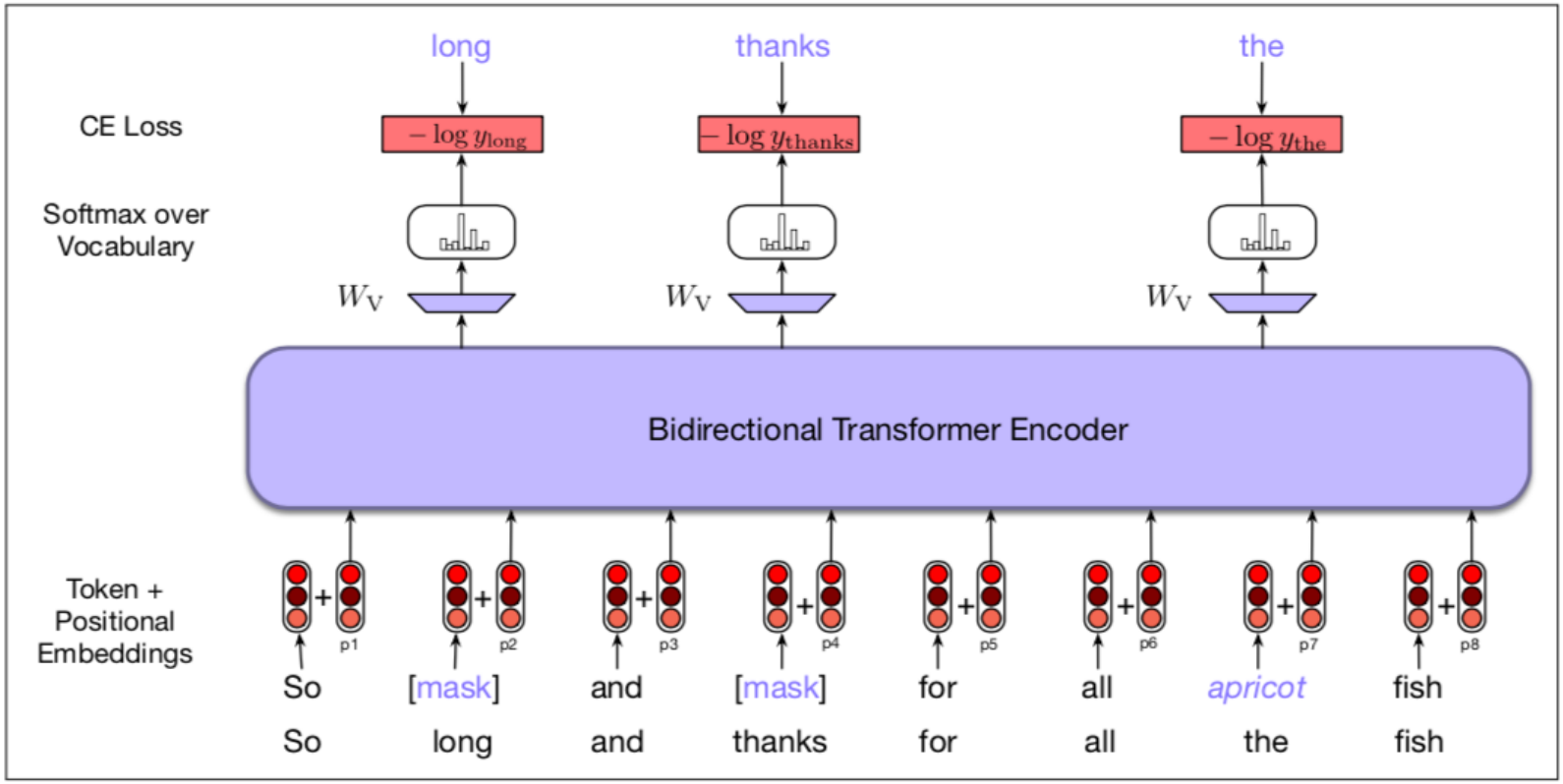
Example : Please turn _____ homework in.

The approach is called **masked language modeling** (MLM).
A random sample of 15% of the input tokens are

- replaced with the unique vocabulary token `[MASK]` (80%)
- replaced with another token randomly sampled with unigram probabilities (10%)
- left unchanged (10%)

Second bullet: used for transfer learning, presented later, in which token `[MASK]` does not appear.

Masked language modeling



Next sentence prediction

Many important downstream tasks involve relationship between pairs of sentences:

- **paraphrase detection**: detecting if two sentences have similar meanings, also called paraphrase identification
- **semantic textual similarity**: detecting the degree of semantic similarity between two sentences
- **sentence entailment**: detecting if the meanings of two sentences entail or contradict each other
- **answer sentence selection**: detecting if the answer to a question sentence is contained in the second sentence

This is not directly captured by language modeling.

Next sentence prediction

In order to train a model that understands sentence relationships, BERT introduces a second learning objective called **next sentence prediction** (NSP).

NSP is a **binary decision** task that can be trivially generated from any monolingual corpus.

NSP produces **sentence embeddings** that can be used for tasks involving relationship between pairs of sentences.

Next sentence prediction

In NSP, the model is presented with pairs of sentences with

- token `[CLS]` prepended to the first sentence
- token `[SEP]` placed between the two sentences and after the rightmost token of the second sentence
- **segment embeddings** marking the first and second sentences are added to each word and positional embeddings

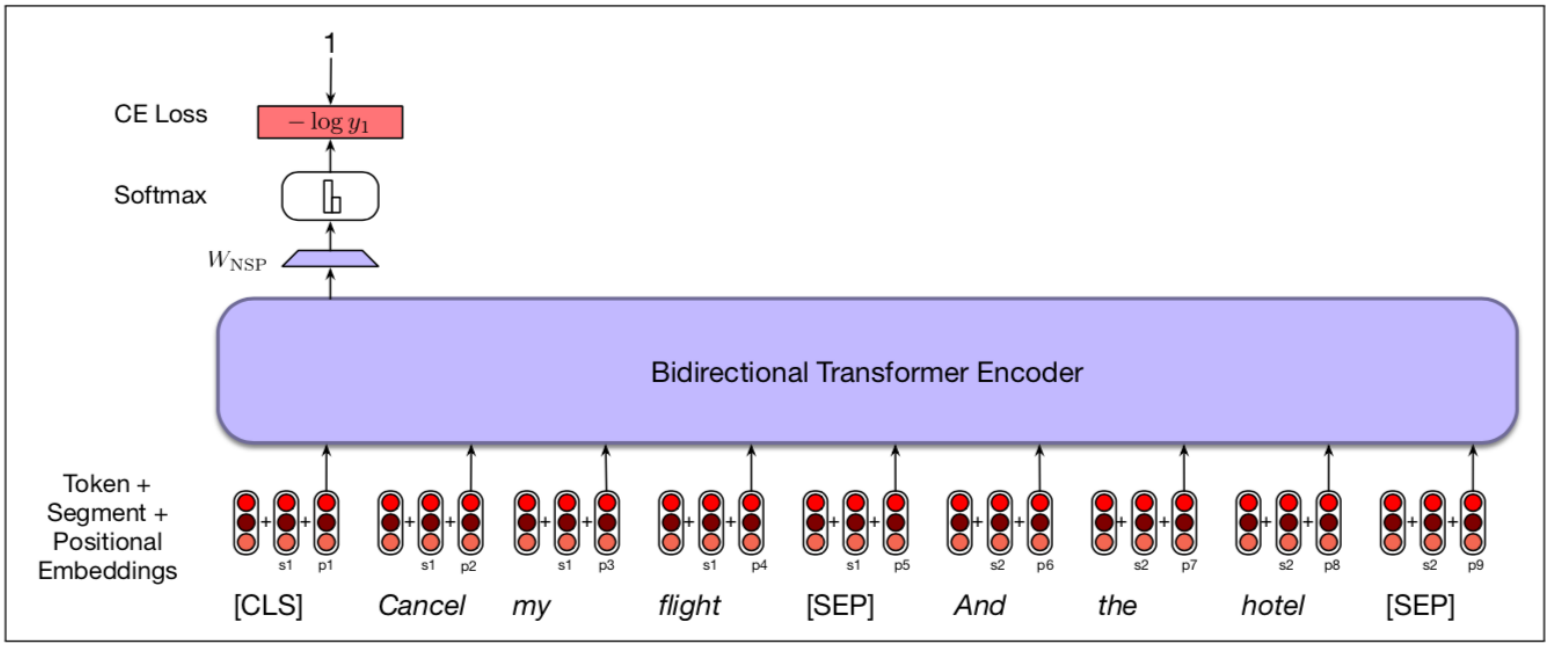
Each pair consists of

- an actual pair of adjacent sentences from the training corpus (50%)
- a pair of unrelated sentences randomly selected (50%)

  Used as in contrastive learning.

Next sentence prediction

The output embedding associated with the [CLS] token represents the next sentence prediction.



Practical issues

The result of the MLM and NSP pre-training processes consists of the **encoder**, which is used to produce contextual embeddings for tokens in novel sentences.

Contrast with static word embeddings.

It is common to compute a representation for token w_t by **averaging** the output embeddings at t from each of the topmost layers of the model.

`[CLS]` embedding also used as **sentence embedding** for tasks involving single sentence classification.

Other models

BERT was significantly undertrained.

RoBERTa (Robustly Optimized BERT Approach) is based on a novel recipe for training BERT

- longer training over more data, with larger batches
- removing NSP pre-training objective
- dynamically changing the masking pattern

  BERT pre-training data are $\times 10$ and masked once for all
- using a byte level BPE tokenizer with larger vocabulary

Several more variants of BERT: ALBERT, BART, SpanBERT, SBERT, mBERT.

Other models

Other popular contextualized word embeddings for English

- CoVe (Context Vectors), one of the very first contextualized word embeddings

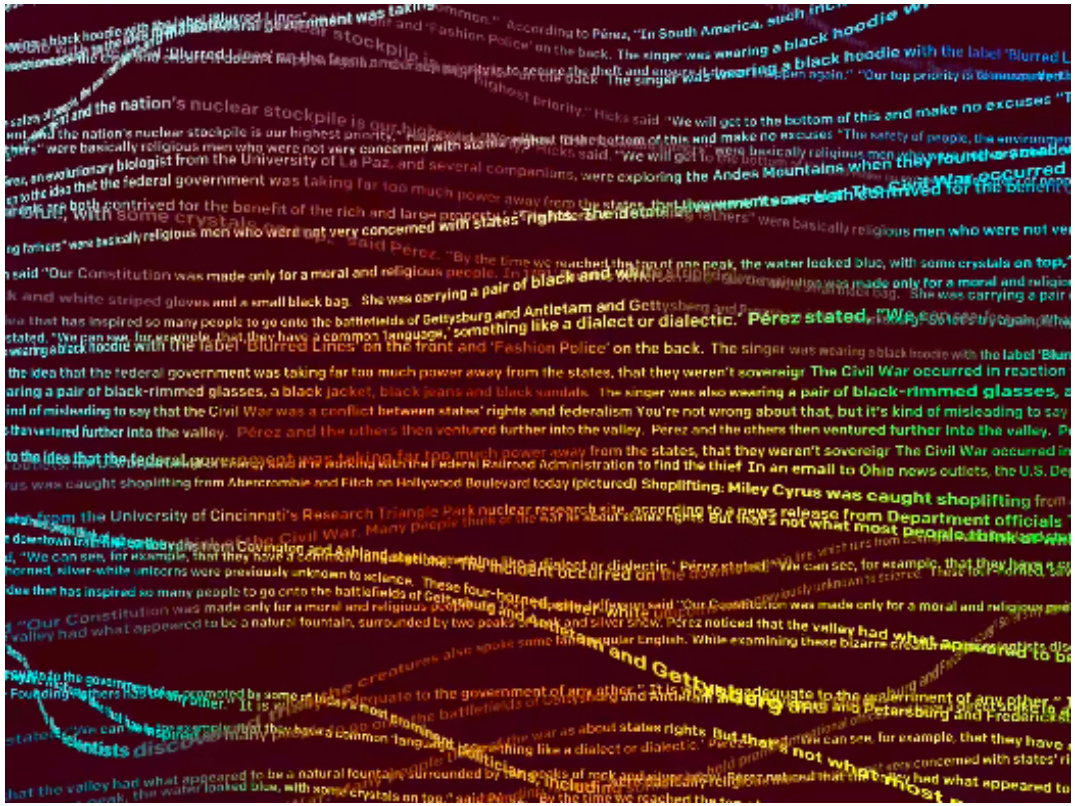
  Based on LSTM, mostly of historical interest.

- ULM-FiT (universal language model fine-tuning) extends CoVe by adding fine-tuning

  First example of fine-tuning / transfer learning in NLP; see second part of this lecture.

- XLNet, a generalized autoregressive pre-training method that overcomes some limitations of BERT
- ELECTRA, a novel pre-training approach which trains two transformer models, called generator and discriminator

Large language model



©OpenAI

Large language models

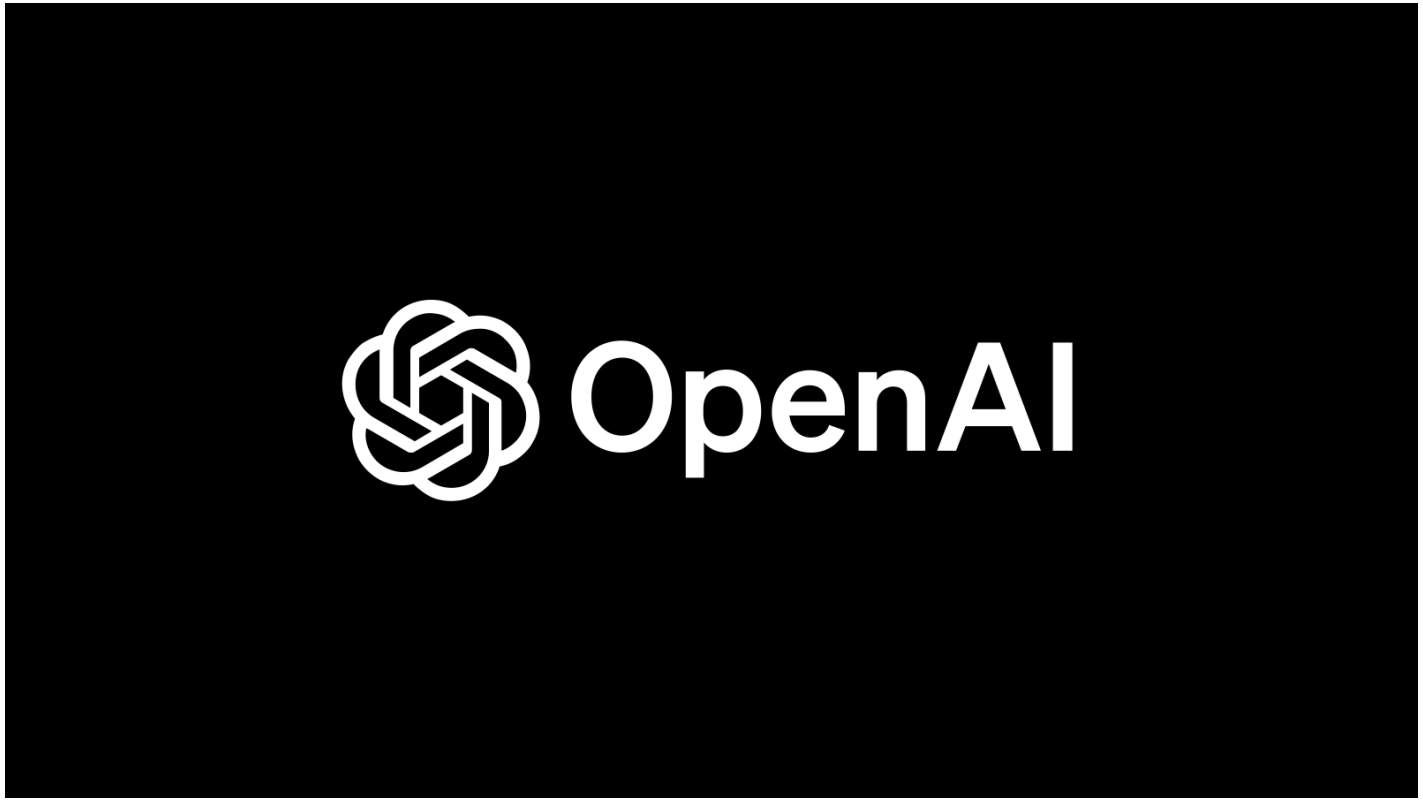
A language model is called **large language model** (LLM) if it has the following features

- neural network architecture based on transformer
- number of parameters larger than 1 billion
- pre-trained on vast amount of textual data
- can be adapted to a wide range of downstream tasks

Sometimes also called foundational models.

LLMs are at the basis of modern applications of **generative artificial intelligence**.

GPT-*n*



GPT-2

GPT-2 (generative pre-training transformer) is a left-to-right (causal) language model.

Created by researchers at OpenAI, 2019.

Based on transformer's decoder-only, no cross-attention layer.
Layer normalization moved to the input of each sub-block.
Tokenization: BPE with vocabulary size from 32,000 to 64,000.

Parameters	Layers	d_{model}
117M	12	768
345M	24	1024
762M	36	1280
1542M	48	1600

GPT-2

Example : Newspaper article generated by GPT-2

Human

In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

GPT-2 (continued)

The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pèrez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pèrez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow [...]

GPT-3

GPT-3 has the same basic architecture as GPT-2: transformer's decoder-only, including pre-normalization.

Created by researchers at OpenAI, 2020.

The model adds several optimization techniques, including (causal) **attention sparsity** at half of the layers, computing only a subset of the elements of the attention matrix.

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size
GPT-3 Small	125M	12	768	12	64	0.5M
GPT-3 Medium	350M	24	1024	16	64	0.5M
GPT-3 Large	760M	24	1536	16	96	0.5M
GPT-3 XL	1.3B	24	2048	24	128	1M
GPT-3 2.7B	2.7B	32	2560	32	80	1M
GPT-3 6.7B	6.7B	32	4096	32	128	2M
GPT-3 13B	13.0B	40	5140	40	128	2M
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M

GPT-3

GPT-3 has been trained on

- Common Crawl dataset (nearly a trillion words)
- WebText dataset
- book corpora
- English-language Wikipedia

Because of such a large textual collection, novel capabilities of LLMs emerge, collectively called **in-context learning**.

Examples in next slide.

GPT-3

Few-shot learning is the ability of an LLM to solve a problem on the basis of a few examples.

Example :
sea otter ⇒ loutre de mer
peppermint ⇒ menthe poivrée
cheese ⇒ ??

Zero-shot learning is the ability of an LLM to solve a new problem solely on the basis of natural language instructions

Example :
translate English to French
cheese ⇒ ??

GPT-3

Using few-shot and zero-shot settings, evaluation has been carried out on

- question answering
- machine translation
- reading comprehension
- common sense reasoning

An enhanced version of GPT-3 named GPT-3.5 has been used as a basis for chatBot chatGPT.

GPT-4

GPT-4 is a **multimodal** large language model.

OpenAI, initially released on March 14, 2023. Rumors claim that the model has 1.76 trillion parameters.

The model is capable of processing image and text inputs and producing text outputs.

The model was then fine-tuned using Reinforcement Learning from Human Feedback (RLHF).

We will see this in detail in the next lecture.

GPT-4

Two versions, with context windows of 8,192 and 32,768 tokens.

On a suite of traditional NLP benchmarks, GPT-4 outperforms most state-of-the-art systems. Also evaluated on a variety of exams originally designed for humans.

When instructed to do so, GPT-4 can interact with **external interfaces**.

A large focus of the GPT-4 project was building a deep learning stack that scales **predictably**: for very large training it is not feasible to do extensive model-specific tuning.

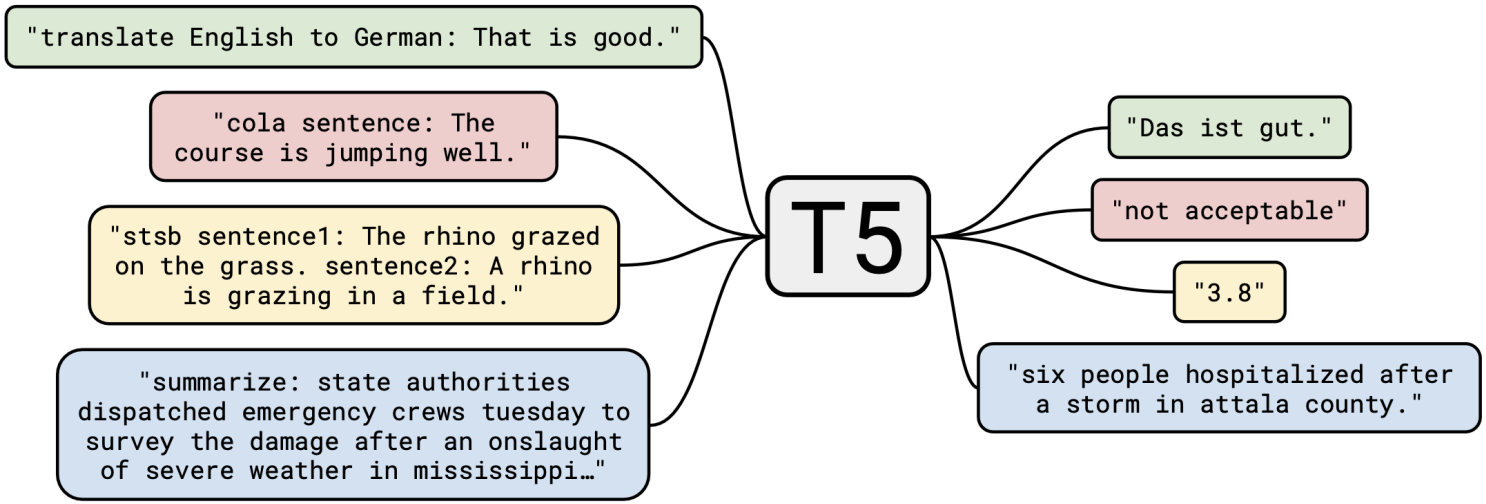
Other LLMs

Rohit Tandon on Unsplash

T5 (text-to-text transfer transformer) is an encoder-decoder based on the transformer architecture.

Created by researchers at Google, 2020.

T5 recasts text-based language problems into text-to-text format.

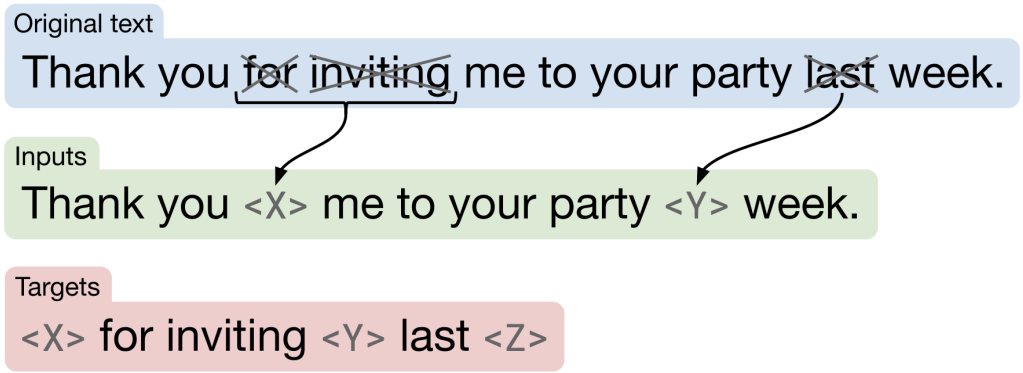


T5

T5 uses a **denoising** objective function, which generalises MLM:

- random substrings of tokens are masked
- replaced by a single token called sentinel, which is **unique** over the example
- the entire sentence must then be reconstructed

Example :



T5 achieves state-of-the-art results on many NLP benchmarks such as summarization, question answering, and text classification.

OPT

OPT (open pre-trained transformer) suite of decoder-only pre-trained transformers ranging from 125M to 175B parameters.

Developed by researchers at Meta AI, 2022.

OPT-175B is comparable to GPT-3, while requiring only (1/7)-th of the carbon footprint to develop.

First time for a language technology system of this size to release the code needed to train and to use the model.

Gopher & PaLM

Gopher is an autoregressive (decoder-only) language model smaller but significantly more accurate than some ultra-large language software.

Created by researchers at DeepMind, 2021.

PaLM (pathways language model) is an encoder-decoder architecture using pathways, a new ML system which enables highly efficient training across multiple TPU Pods.

Developed by researchers at Google, 2022.

LaMDA

LaMDA (Language Models for Dialog Applications) is a family of conversational large language models.

Developed by Google in 2022 with 137 bilion parameters, powering BARD (chatBot).

In June 2022, LaMDA gained widespread attention when a Google engineer made claims that the chatBot had become sentient.

Orca

Orca: aiming at enhancing the capability of smaller models through imitation learning, drawing on the outputs generated by larger LM.

Developed by Microsoft and released in 2023, 13B parameters.

Orca learns from rich signals from GPT-4, including explanation traces, step-by-step thought processes, and other complex instructions.

Other LLMs

After the release of GPT-3, actors from several large countries other than the US have joined the race

- BAAI releases Chinese pre-trained language model (CPM)
- Huawei (China) announces Pangu-α
- Naver (South Korea) unveils HyperCLOVA
- BAAI announces WuDao 2.0
- AI21 Labs (Israel) announces Jurassic-1
- Peng Cheng Laboratory and Baidu (China) release PCL-BAIDU Wenxin

`https://lastweekin.ai/p/gpt-3-foundation-models-and-ai-nationalism`

Open LLMs

LLaMA (Large Language Model Meta AI) is a family of decoder-only LLMs released by Meta AI starting in February 2023.

Model weights have been released to the research community under a noncommercial license.

Four model sizes were trained: 7, 13, 33 and 65 billion parameters, last one trained on 1.4 trillion corpus size.

LLaMA-2 is the next generation of LLaMA. The model architecture remains largely unchanged, but 40% more data was used to train.

Three model sizes: 7, 13, and 70 billion parameters. LLaMA-2 7B version has been used as a basis for the freely distributed chatBot Vicuna.

Open LLMs

Falcon 7B is a causal decoder-only LLM built by Technology Innovation Institute (TII) in 2023. It is made available under the Apache 2.0 license.

Model has 7 billion parameters. Trained on 1,500B tokens of RefinedWeb, enhanced with curated corpora.

Mistral 7B is a decoder-based LLM released by Mistral AI in 2023 in fully open version.

Model has 7.3 billion parameters. Size of training data is unknown at moment of writing.

It outperforms Llama2 13B on all benchmarks.

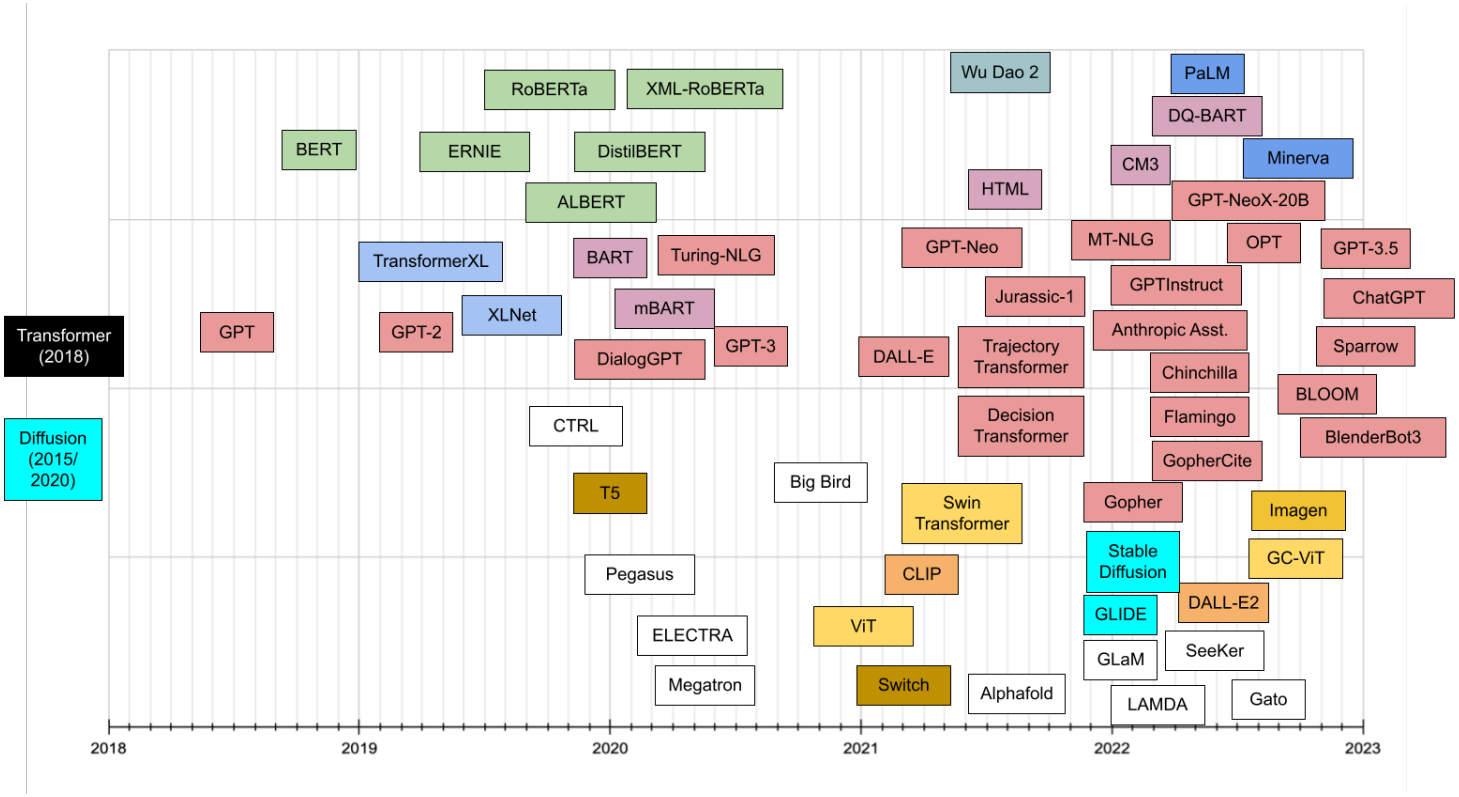
Open LLMs

Solar https://ollama.ai/library/solar

Mixtral https://mistral.ai/news/mixtral-of-experts/

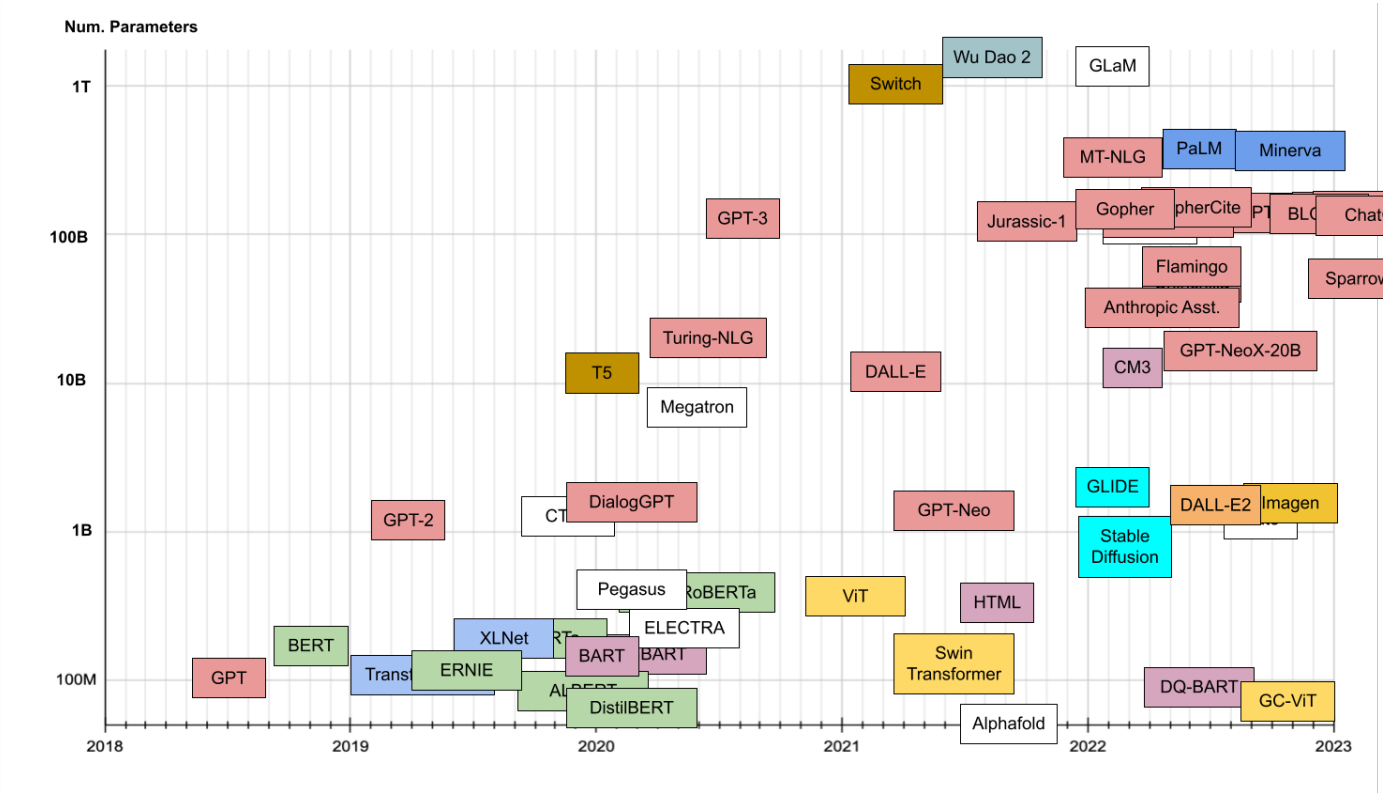
OLMo ...

A timeline for LLMs



https://amatriain.net/blog/transformer-models-an-introduction-and-catalog-2d1e9039f376/

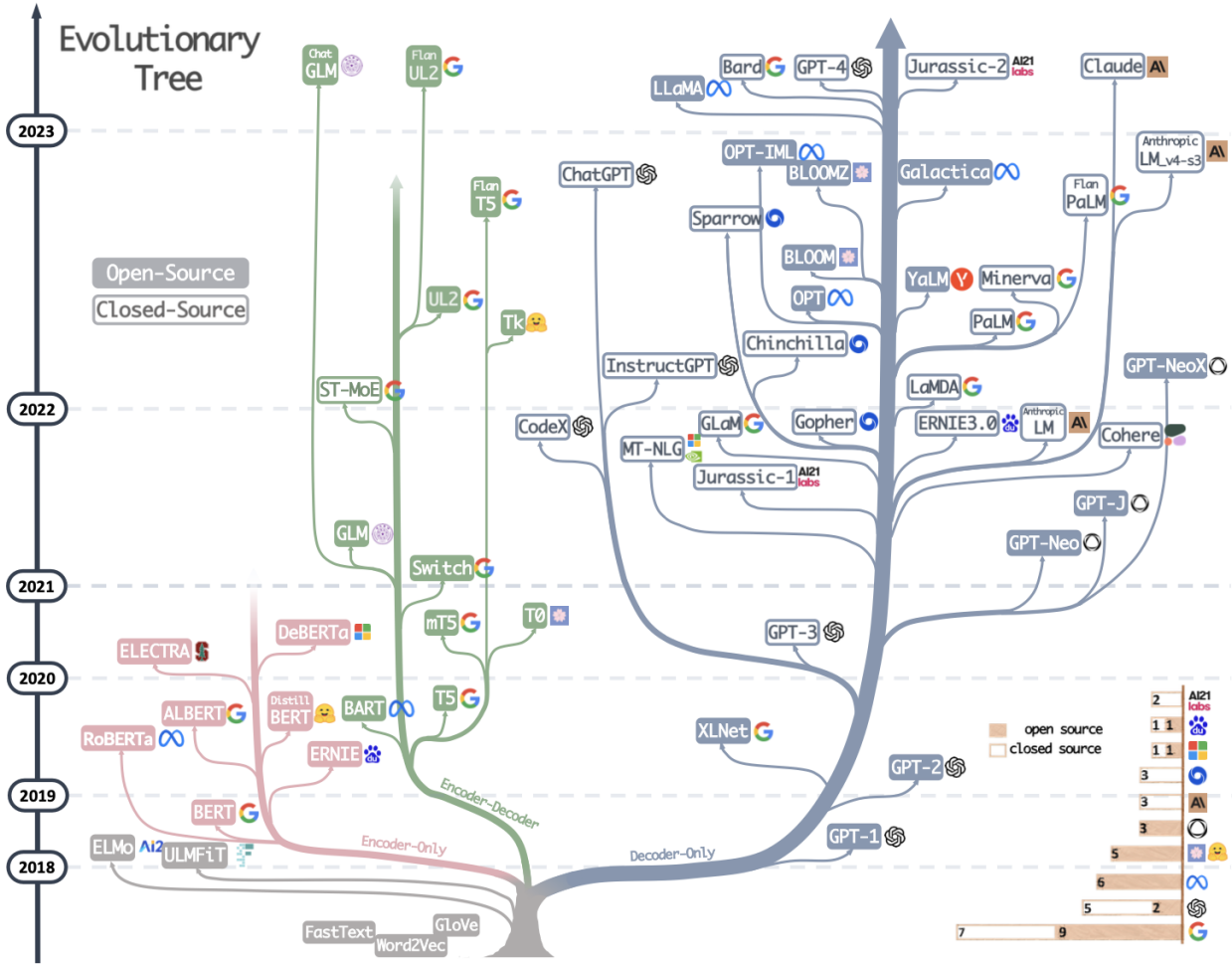
LLMs and number of parameters



https://amatriain.net/blog/transformer-models-an-introduction-and-catalog-2d1e9039f376/

Overview

Classification for LLMs: encoder-only, decoder-only, encoder-decoder.



Yang et al., Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond 3

Multi-lingual LLMs

Oskar Kadaksoo on Unsplash

Multi-lingual LLMs

Multilingual large language models (MLLMs) are typically based on the encoder part of the transformer architecture.

A MLLM is pretrained using large amounts of unlabeled data from multiple languages, ranging from 10 to 100.

Very useful in **transfer learning** (see second part of lecture), where low resource languages can benefit from high resource languages.

Multi-lingual LLMs

Most popular MLLM

- mBERT (multimingual BERT) and several variants
- XLM (cross-lingual language model)
- mT5 (multilingual T5)
- BLOOM, the largest multilingual open-source model to date

  Trained using data from 46 languages; developed in 2022 by several groups, including HuggingFace, Microsoft, NVIDIA and PyTorch

Training of MLLMs

During pre-training, MLLMs use two different sources of data

- large monolingual corpora in individual languages
- parallel corpora between some languages

Objective functions can be based on monolingual data alone by

- pooling together monolingual corpora from multiple languages
- applying masked language modeling (MLM)

Surprisingly, the encoder learns word representations across languages that are **aligned** (close vectors), without the need for any parallel corpora.

Training of MLLMs

Objective functions based on parallel corpora **explicitly** force representations of corresponding words across languages to be close to each other in the multilingual encoder space.

Using MLM, to predict a masked word `[MASK]` for language A the system can

- rely on words for language A surrounding `[MASK]`
- rely on an **aligned** word that has not been masked, that is, a word representing a translation in language B of the masked word

Training of MLLMs

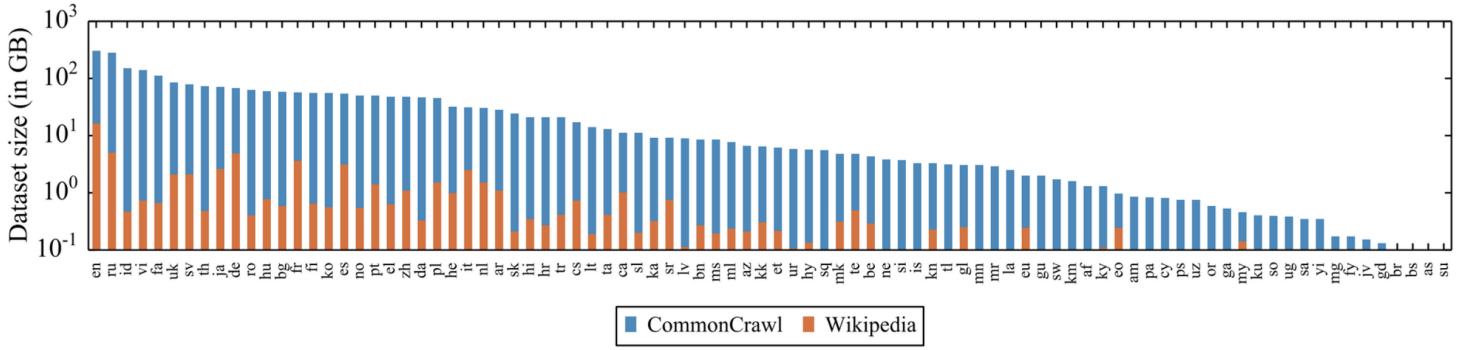
Since parallel corpora are generally much smaller than monolingual data, the parallel objectives are often used in conjunction with monolingual models.

Curse of multilinguality: Similar to models that are trained on many tasks, the more languages a model is pre-trained on, the less model capacity is available to learn representations for each language.

Multi-lingual LLMs

English language is overrepresented.

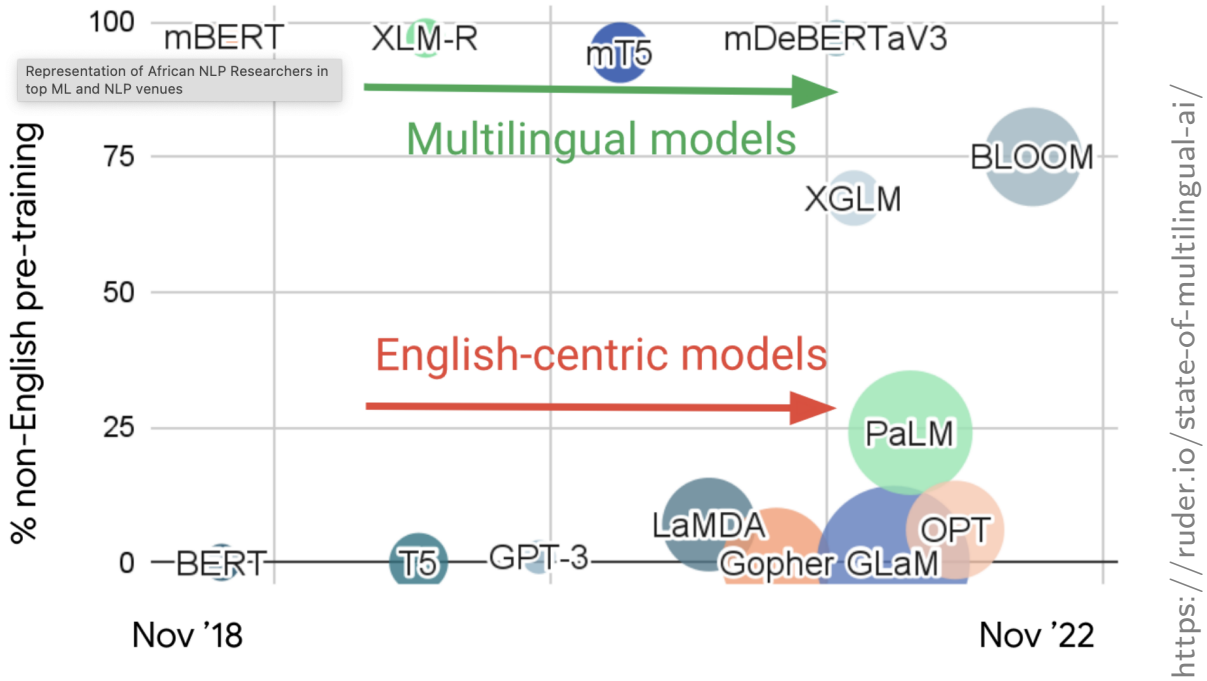
Amount of data in GB (log-scale) for the 88 languages that appear in CommonCrawl and Wikipedia.



Conneau *et al.*, 2020

Multi-lingual LLMs

The largest recent models are not becoming significantly more multilingual.



SBERT

Samuel Ferrara on Unsplash

SBERT

In some sentence-pair regression tasks, training BERT on all sentence pairs is computationally unfeasible.

Examples: semantic textual similarity, answer sentence selection.

Alternatively individual **sentence embeddings** can be computed

- by averaging the BERT output layer, known as BERT embeddings, or
- by using the output of the `[CLS]` token

The above methods result in **bad** sentence embeddings, often worse than averaged GLoVE embeddings.

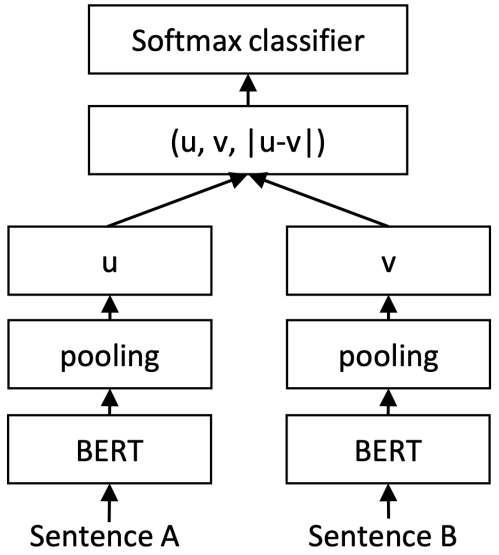
SBERT

Sentence-BERT, or SBERT for short, is an efficient and effective method to compute sentence embeddings.

SBERT uses a **siamese neural network**, a special network that contains two identical subnetworks sharing their parameters.

Siamese networks are most commonly used to compute **similarity scores** for the inputs, and have many applications in computer vision.

SBERT training



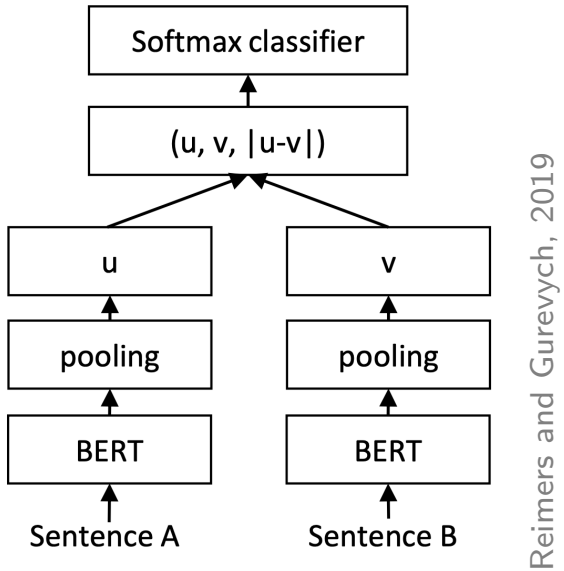
Reimers and Gurevych, 2019

SBERT concatenates vectors u, v, and the element-wise difference $|u - v|$ and uses trainable matrix $\mathbf{W} \in \mathbb{R}^{3n \times k}$ to compute

$$o = \text{softmax}(\mathbf{W}[u; v; |u - v|])$$

with n dimension of the embeddings and k number of labels.

SBERT training



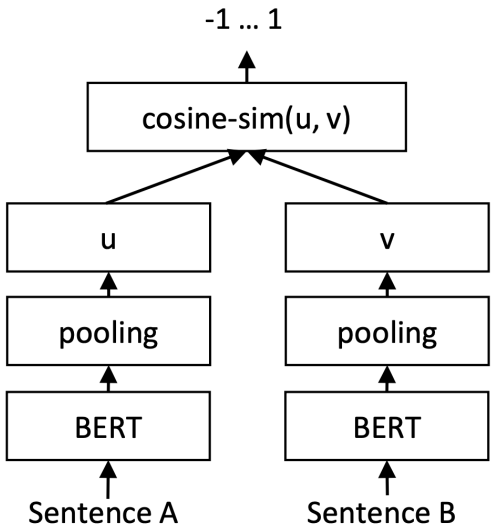
Reimers and Gurevych, 2019

BERT is **fine-tuned** (see second part of lecture).

Parameter updating is mirrored across both sub-networks.

Optimization uses cross-entropy loss.

SBERT inference



Reimers and Gurevych, 2019

The two embeddings are compared using a cosine similarity function, which will output a similarity score for the two sentences.

Miscellanea

Kristine Rosenblatt, Kristine's Kitchen

Emergent abilities

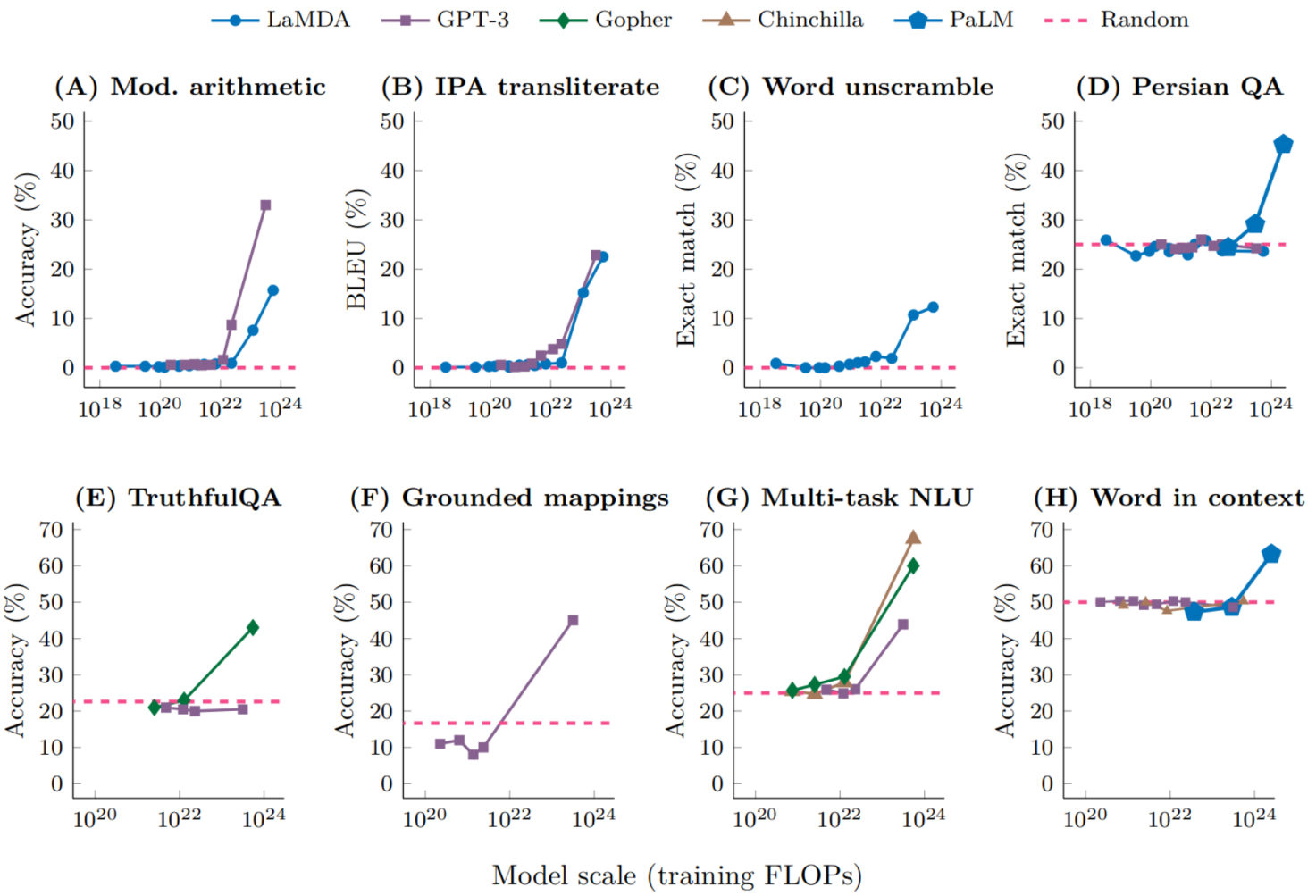
Emergent abilities of large language models are characterized by

- sharpness: transitioning seemingly instantaneously from not present to present
- unpredictability: transitioning at seemingly unforeseeable model scales

This issue is still under debate and controversial.

It has been argued that emergent abilities might be induced by wrong choice of metrics.

Emergent abilities



Wei et al., Emergent abilities of large language models, 2022

Hallucinations

Hallucination refers to a phenomenon where a LLM generates text that is incorrect, nonsensical, or not real.

Since LLMs are not databases or search engines, they would not cite where their response is based on.

Since LLMs need to be creative, they must be able to invent scenarios that are not factual.

To avoid hallucinations use **controlled generation**: providing enough details and constraints in the prompt to the model.

Prompting will be presented later on in this lecture.

Mixture of Experts

Mixture of Experts (MoE) is a machine learning technique where multiple expert networks (learners) are used to divide a problem space into homogeneous regions.

MoE differs from ensemble techniques in that typically only one models will be run, rather than combining results from all models.

This technique has been recently used in LLMs. The result is a sparsely-activated model with a large number of parameters but a constant computational cost.

Research papers

Emil Widlund on Unsplash

Research papers

Title: Is it possible for language models to achieve language understanding?
Authors: Christopher Potts
Source: October 5, 2020
Content: This post is mainly devoted to critically reviewing prominent arguments that language models are intrinsically limited in their ability to understand language.

`https://chrisgpotts.medium.com/is-it-possible-for-language-models-to-achieve-language-understanding-81df45082ee2`

Primer

Title: What We Know About LLMs
Authors: Will Thompson
Source: July 23, 2023
Content: A good summary of the topics in this lecture, along with other research lines.

`https://willthompson.name/what-we-know-about-llms-primer`

Primer

Title: The Practical Guides for Large Language Models
Authors: Will Thompson
Source: August 6, 2023
Content: A curated (still actively updated) list of practical guide resources of LLMs.

`https://github.com/Mooler0410/LLMsPracticalGuide`

Fine-tuning

Joel Wyncott from Unsplash

Adaptation

To make practical use of LLMs, we need to interface these models with downstream applications.

This process is called **adaptation**; it uses supervised learning (labeled data) for the task of interest.

Adaptation is the prevalent paradigm today in natural language processing.

Adaptation

Two most common forms of adaptation are

- **feature extraction** (also called feature-based transfer): freeze the pre-trained parameters of the language model and train parameters of the model for the task at hand.

  We use the hidden states as features and just train a classifier on them.

- **fine-tuning**: make (possibly minimal) adjustments to the pre-trained parameters.

Retraining the whole model (all parameters) results in so-called **catastrophic forgetting**.

Fine-tuning

Example : Let $\mathbf{z}_{\text{CLS}}$ be the embedding for the token `[CLS]`. For **sentiment analysis** define the classifier

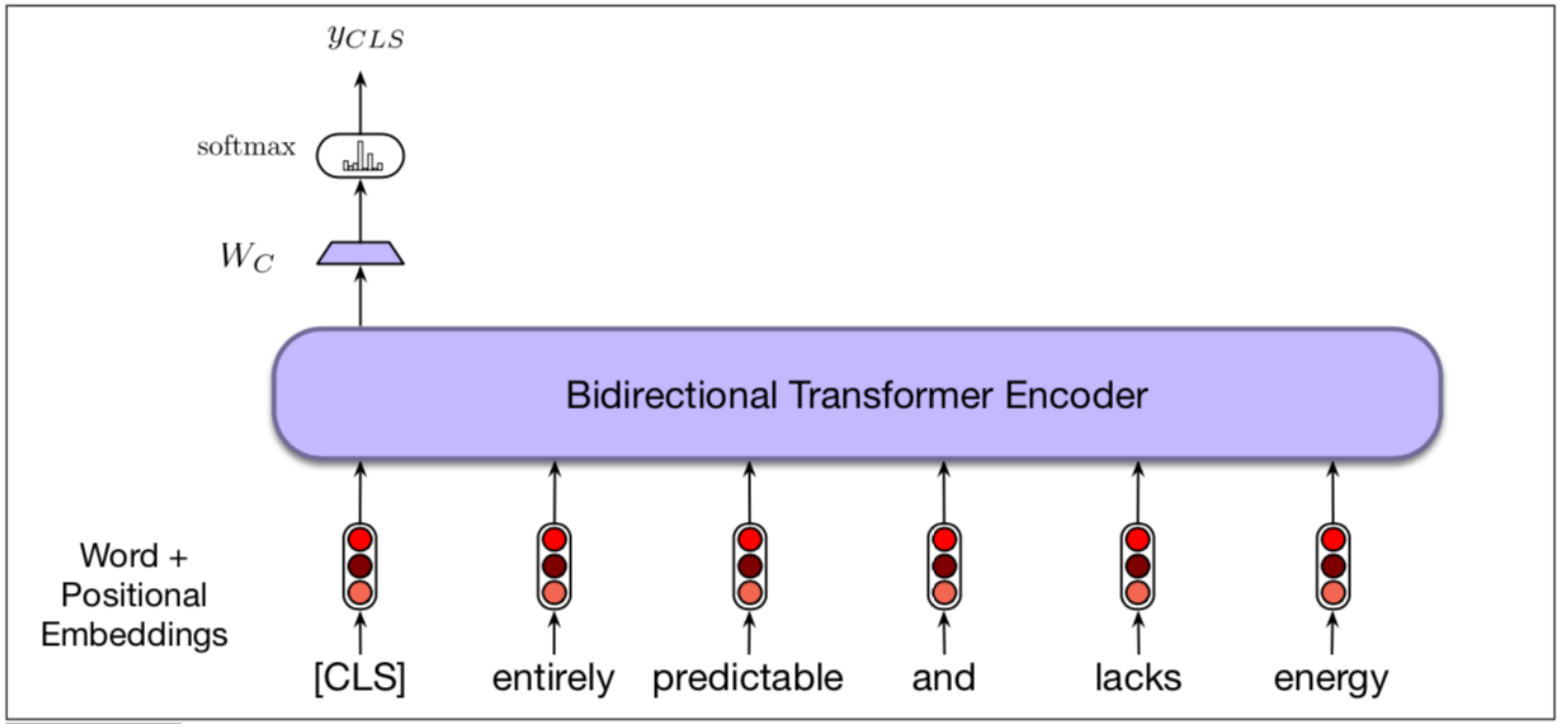
$$\mathbf{y}_{\text{CLS}} = \text{softmax}(\mathbf{W}_C \cdot \mathbf{z}_{\text{CLS}} + \mathbf{b})$$

Use labeled data to learn $\mathbf{W}_C, \mathbf{b}$ and to **update** the weights for the pre-trained language model itself.

In practice, only **minimal** changes to the language model parameters are needed, limited to updates over the final few layers of the transformer.

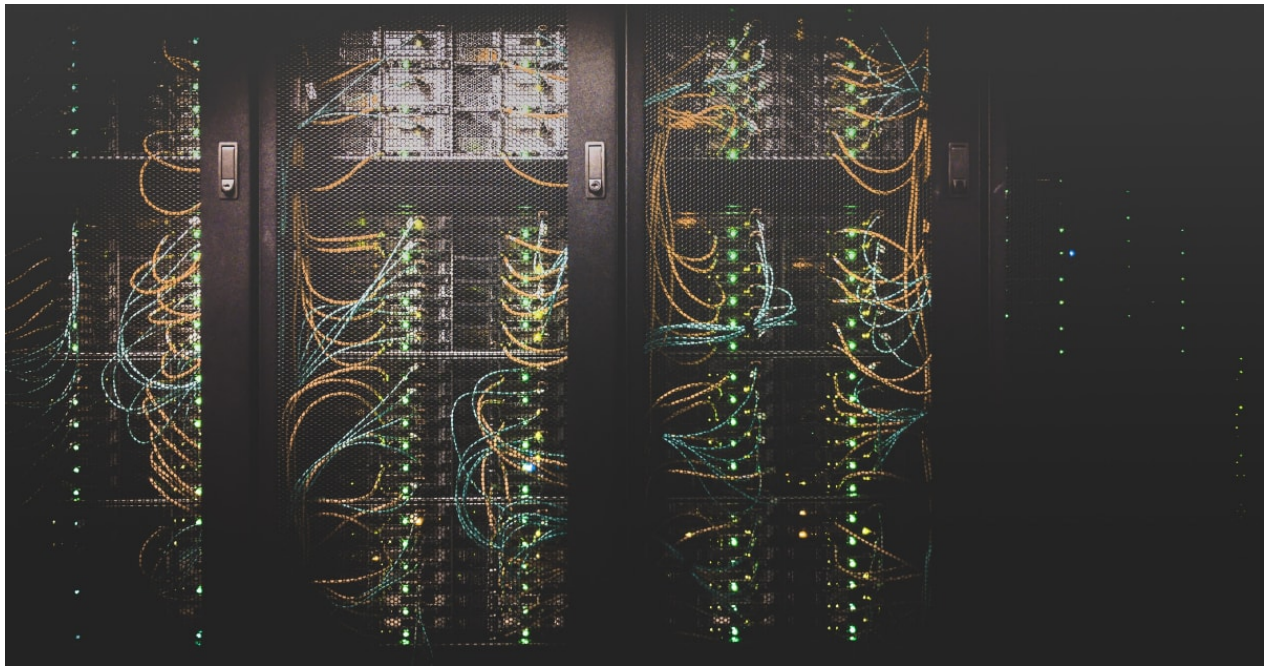
Fine-tuning

Example : Sentiment analysis, only top-most layers of the transformer are updated.



W_C usually called classification head.

Parameter efficient fine-tuning



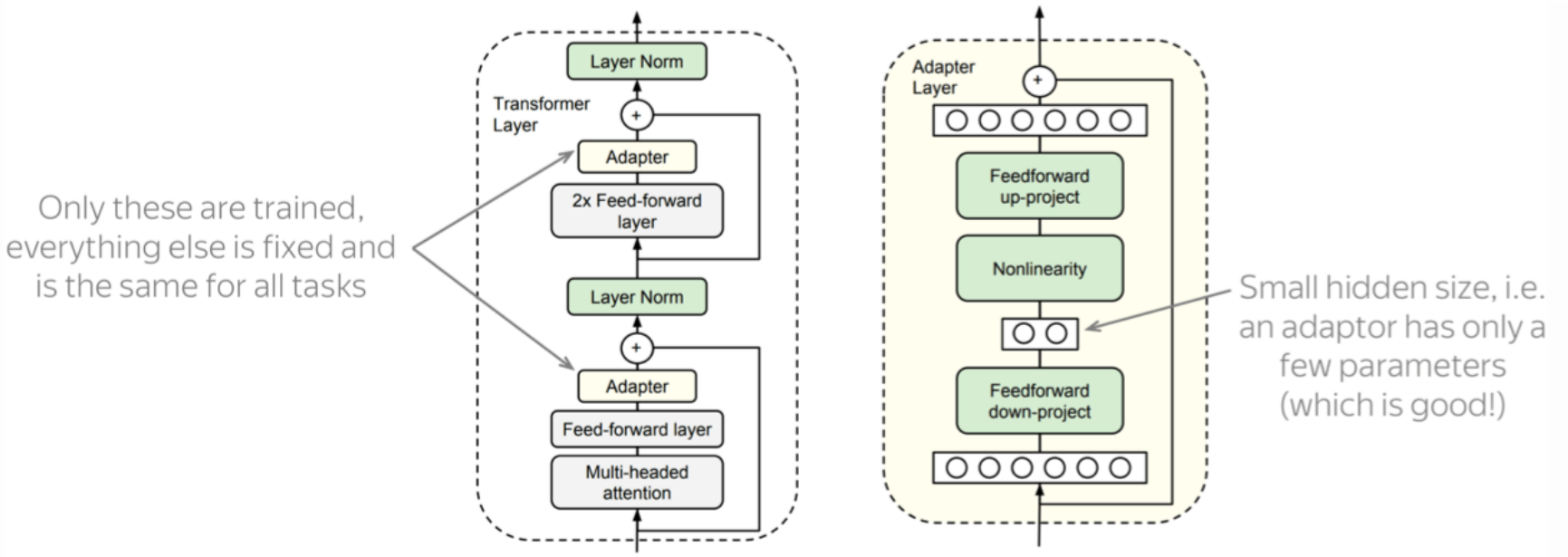
Adapters

When working with huge pre-trained models, fine-tuning may still be **inefficient**.

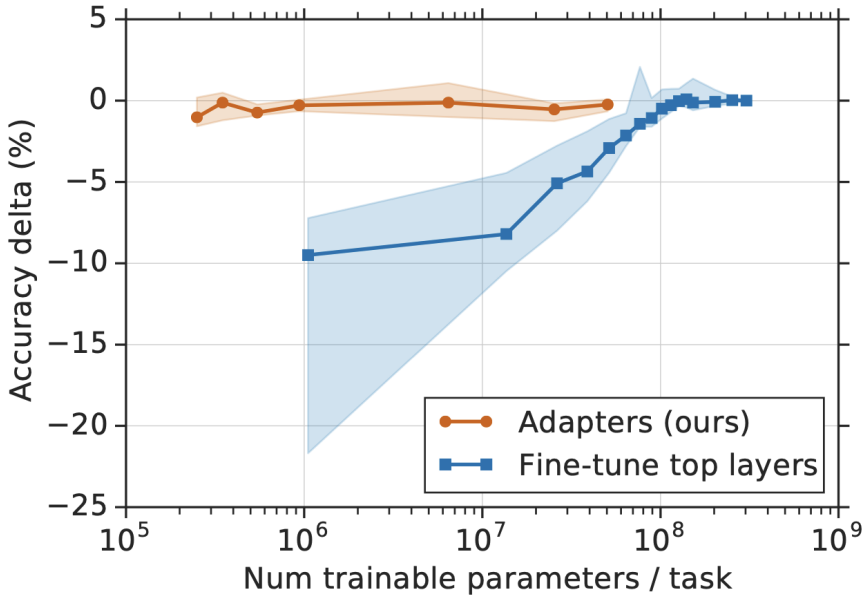
Alternatively, one could fix the pre-trained model, and train only small, very simple components called **adapters**.

With adapter modules transfer becomes very **efficient**: the largest part of the pre-trained model is shared between all downstream tasks.

Transformer with adapter module consisting of a two-layer feed-forward network with a nonlinearity and a residual connection.



Adapter-based tuning attains a similar performance to top-layer fine-tuning, with two orders of magnitude fewer trained parameters.



Houlsby *et al.*, 2019

LoRA

LoRA (Low-Rank Adaptation) is a popular fine-tuning strategy, alternative to adapters. It drastically reduces the number of trainable parameters.

LoRA is based on the idea of **intrinsic dimensions**: there exists a low dimension reparameterization that is as effective for fine-tuning as the full parameter space.

LoRA is known for its high efficiency, and for avoiding catastrophic forgetting.

LoRA

We can think of **weight update** as follows

$$W \leftarrow W + \Delta W$$

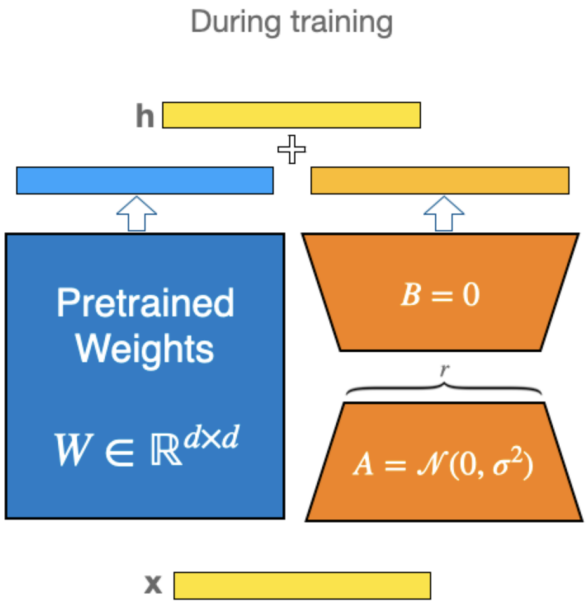
where $W \in \mathbb{R}^{d \times d}$.

In LoRA we update the weights using a decomposition of ΔW into two low-rank matrices

$$\Delta W = BA$$

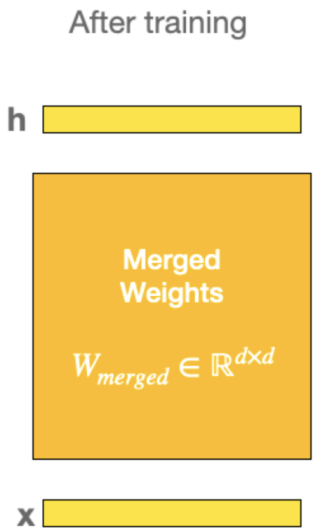
where $B \in \mathbb{R}^{d \times r}$, $A \in \mathbb{R}^{r \times d}$, and r is some low **rank**.

LoRA



$$h = Wx + BAx$$

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



Saurav Maheshkar, A Brief Introduction to LoRA: Low-Rank Adaptation of Large Language Models

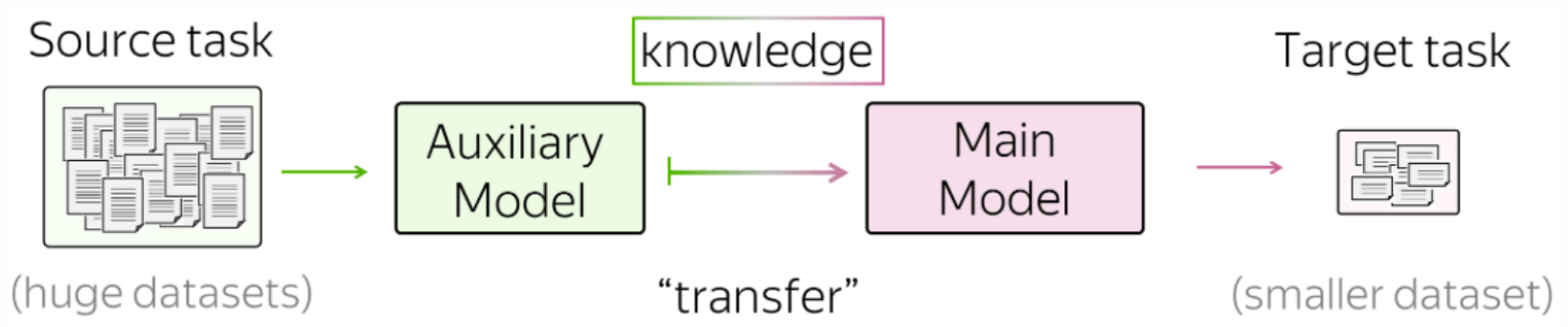
Rhys Moult from Unsplash

Transfer learning

The pre-train/fine-tune paradigm is a special case of a machine learning approach called transfer learning.

The general idea of **transfer learning** is to reuse information from a previously learned source task for the learning of a target tasks.

This is used when you don't have enough data for the target task.

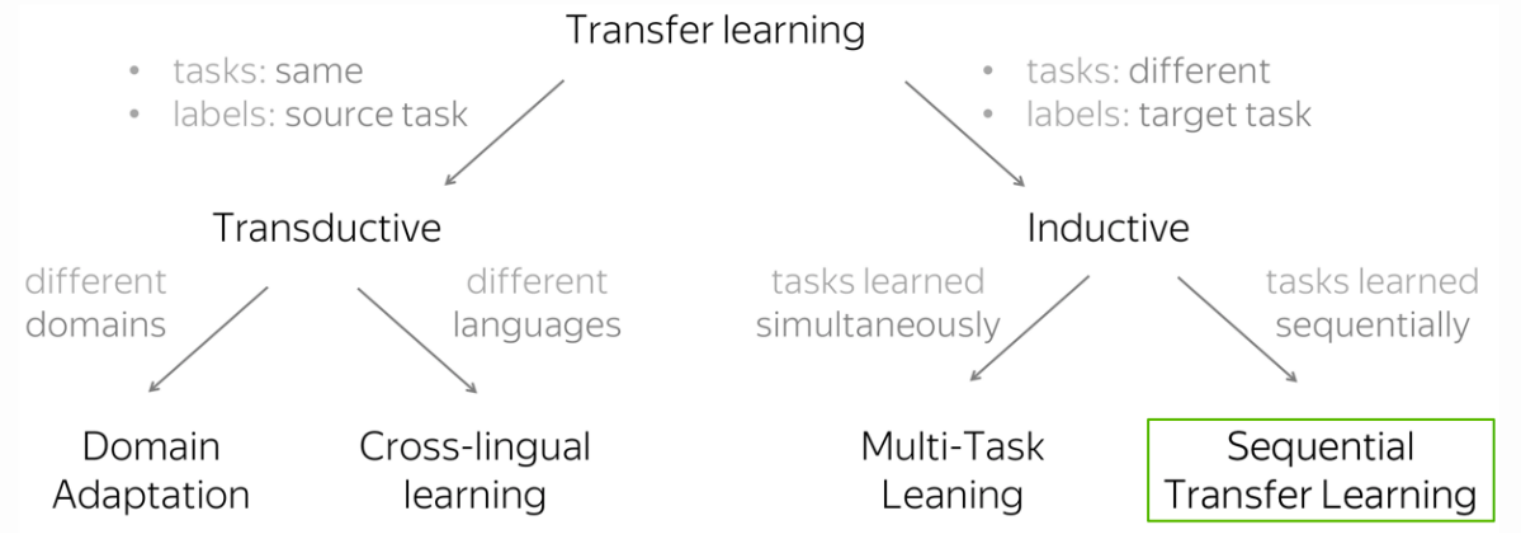


Transfer learning

There are several types of transfer learning

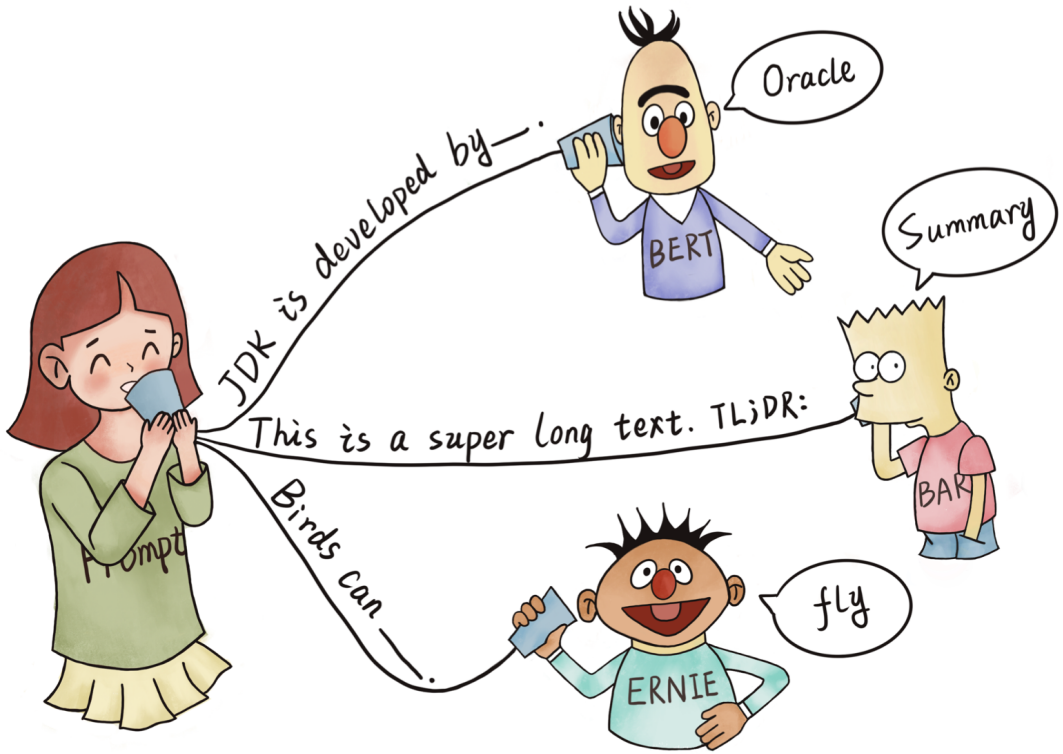
See for instance Sebastian Ruder's blog post at

`https://ruder.io/state-of-transfer-learning-in-nlp/`



Fine-tuning, also known as **sequential transfer learning**, is the most popular transfer learning approach in NLP.

Prompt learning



Liu *et al.*, 2021

Prompt learning

Fine-tuning LLMs on task-specific downstream applications has become **standard practice** in NLP.

However, the GPT-3 model with 175B parameters has brought a new way of using LLMs for downstream tasks.

We can handle a wide range of tasks with only a few examples, by **formulating** the original task as a word prediction problem, while not updating the parameters in the underlying model.

Prompt learning

A **prompt** is a piece of text inserted in the input examples, so that the original task can be formulated as a masked language modeling problem.

Example : When recognizing the emotion of a social media post, we may append to the post itself the following prompt

'I missed the bus today. I felt so _____ .'

and ask the LM to fill the blank with an emotion-bearing word.

Prompt learning

Example : Similarly for machine translation we can append the prompt

'English: I missed the bus today. French: _____.'

and the LM may be able to fill in the blank with a French translation.

Example : For sentiment analysis we can use the prompt

'No reason to watch. It was _____.'

and expect the LM to predict the word 'terrible' with a higher probability than the word 'great'.

Prompt learning

In **prompt learning** we give the LM a few example prompts to demonstrate a task.

We then append a test prompt and ask the LM to make a prediction, **conditionally** to the example prompts.

Prompt learning is also referred to as in-context learning.

The term 'in-context learning' was introduced in the original GPT-3 paper (Brown *et al.*, 2020).

The already mentioned zero-shot/few-shot learning can also be viewed as special cases of prompt learning.

Prompt learning

Example :

Circulation revenue has increased by 5% in Finland. // Positive

Panostaja did not disclose the purchase price. // Neutral

Paying off the national debt will be extremely painful. // Negative

The company anticipated its operating profit to improve. // _____

Positive

Circulation revenue has increased by 5% in Finland. // Finance

They defeated ... in the NFC Championship Game. // Sports

Apple ... development of in-house chips. // Tech

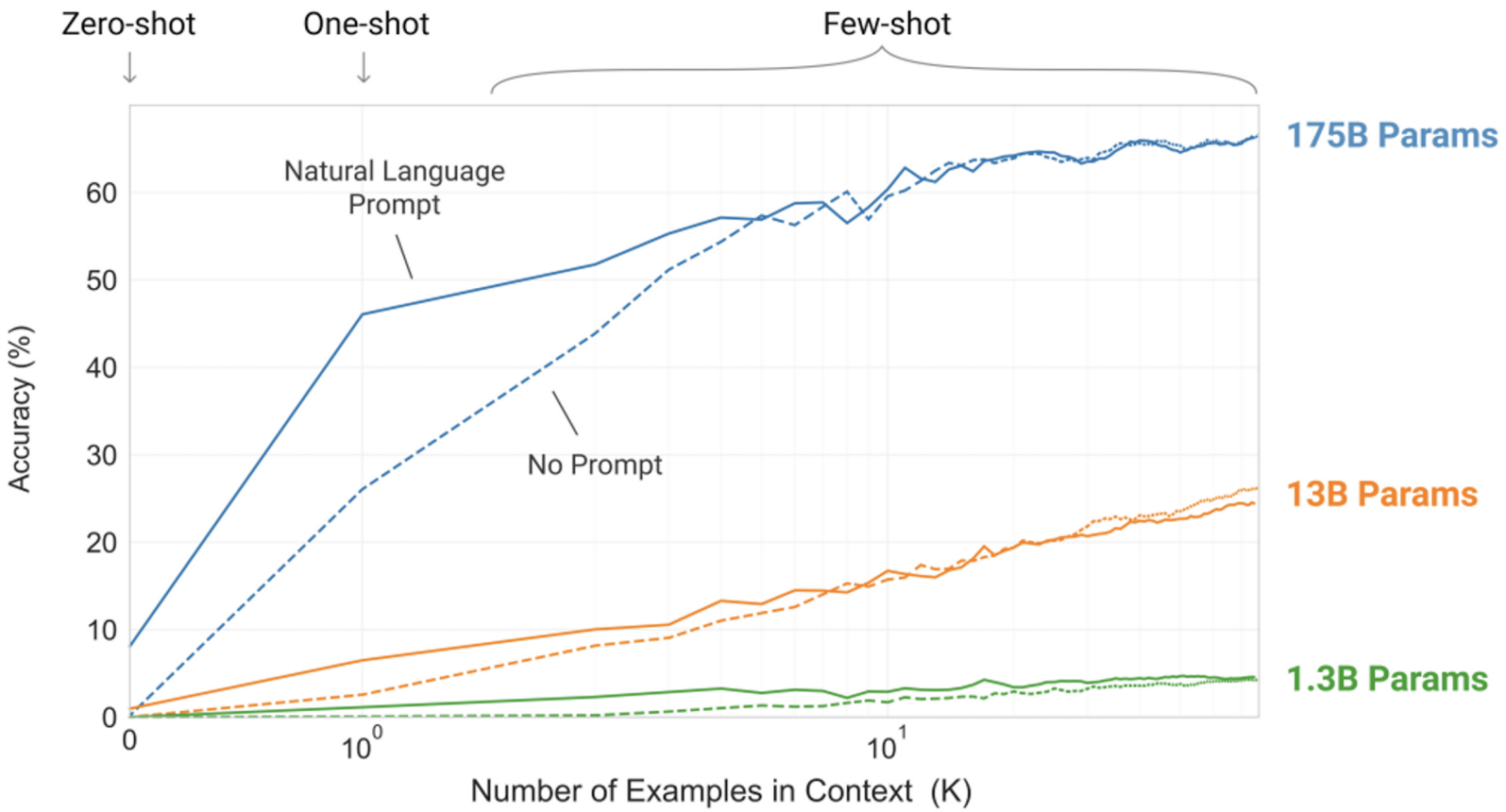
The company anticipated its operating profit to improve. // _____

Finance

https://ai.stanford.edu/blog/understanding-incontext/#f1

Prompt learning

Accuracy with few shot learning for several GPT-3 models:



Prompt learning

The advantage of prompt learning is that, given a suite of appropriate prompts, a single LM trained in an entirely **unsupervised** fashion can be used to solve several tasks.

This method introduces the necessity for **prompt engineering**: finding the most appropriate prompt to allow a LM to solve the task at hand.

Prompt learning

Why does prompt learning work?

- prompt examples do not teach a new task; instead, they help locating a task already learned during pre-training
- prompt examples induce a latent state/concept so that LM generates coherent next tokens
- prompt learning performance is highly correlated with term frequencies during pre-training

Prompt learning

Advantages of prompt learning wrt fine-tuning

- the gap between the pre-training stage and the downstream task can be significant: the objectives are different
- for the downstream tasks, we need to introduce new parameters
- when you only have a dozen of training examples for a new downstream task, it is hard to fine-tune

Prompting can also be used to avoid hallucinations, providing enough details and constraints in the prompt to the model.

Retrieval-augmented generation

©Medium

Retrieval-augmented generation

Retrieval-augmented generation (RAG) is a two-step process combining external knowledge search with prompting.

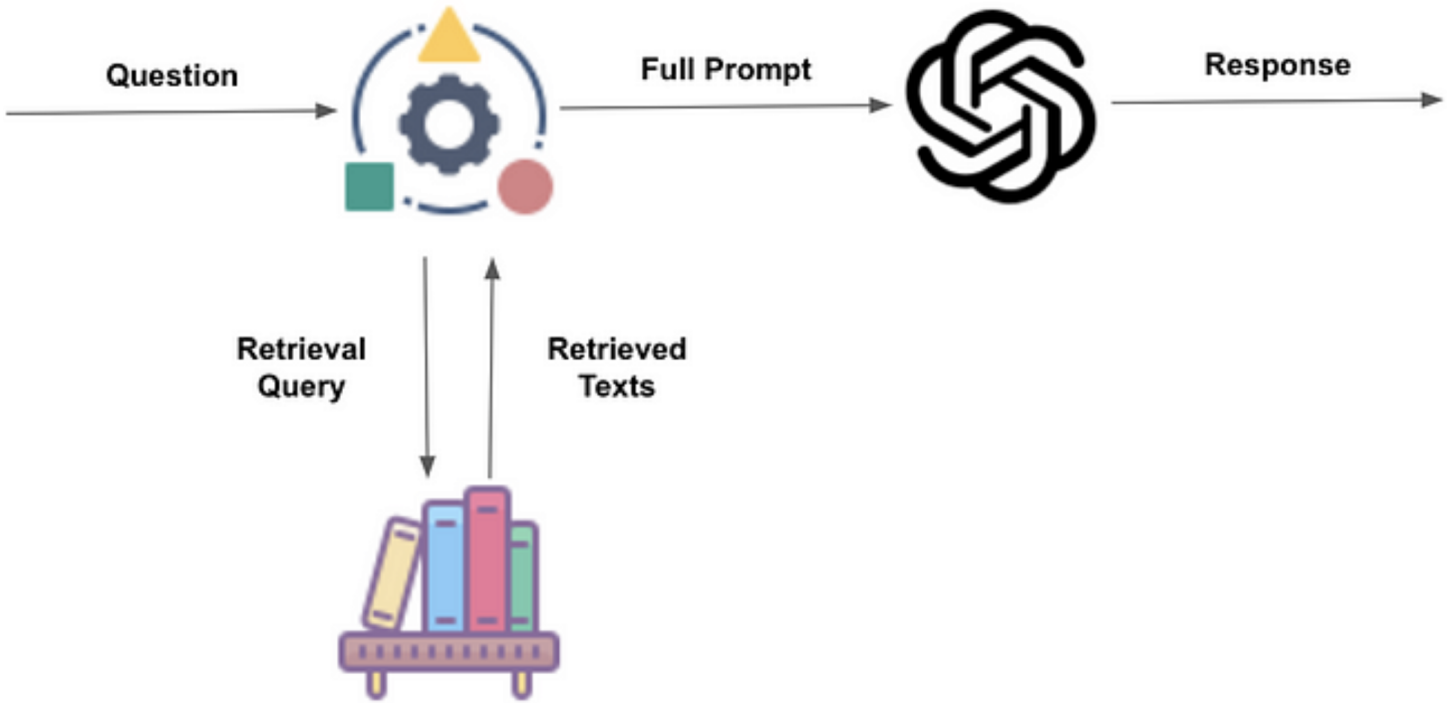
Retrieval: given a query, some information retrieval neural model fetches relevant documents/passages.

Based on document and query embedding, and a learnable similarity measure.

Generation: fetched documents are wrapped into a prompt and passed to the LLM to generate relevant response.

Retrieval-augmented generation

RAG: retrieval and generation steps



Retrieval-augmented generation

Three important advantages of RAG approach.

Flexibility: external knowledge source injects into LLM recent/specific information that wasn't available during pre-training.

Efficiency: RAG works well also with small-size, more manageable LLMs.

Updatability: external knowledge source can be updated, no need to retrain the LLM.

Ethics

Feng Yu, Stock.Adobe.com

Ethics

Contextual language models can generate **toxic language**, misinformation, radicalization, and other socially harmful activities.

Contextual language models can **leak information** about their training data. It is possible for an adversary to extract individual data from a language model (phishing).

Mitigating all these harms is an important but **unsolved** research problem in NLP.

Research papers

Emil Widlund on Unsplash

Research papers

Title: Language Models are Few-Shot Learners
Authors: Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal *et al.*
Conference: NeurIPS 2020
Content: We demonstrate that scaling up language models greatly improves task-agnostic, few-shot performance, sometimes even becoming competitive with prior state-of-the-art fine-tuning approaches.

`https://papers.nips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html`

Research papers

Title: To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks
Authors: Matthew E. Peters, Sebastian Ruder, Noah A. Smith
Workshop: 4th ACL Workshop on Representation Learning for NLP
Content: This work asks how to best adapt the pretrained model to a given target task. It is empirically shown that the relative performance of fine-tuning vs. feature extraction depends on the similarity of the pre-training and target tasks.

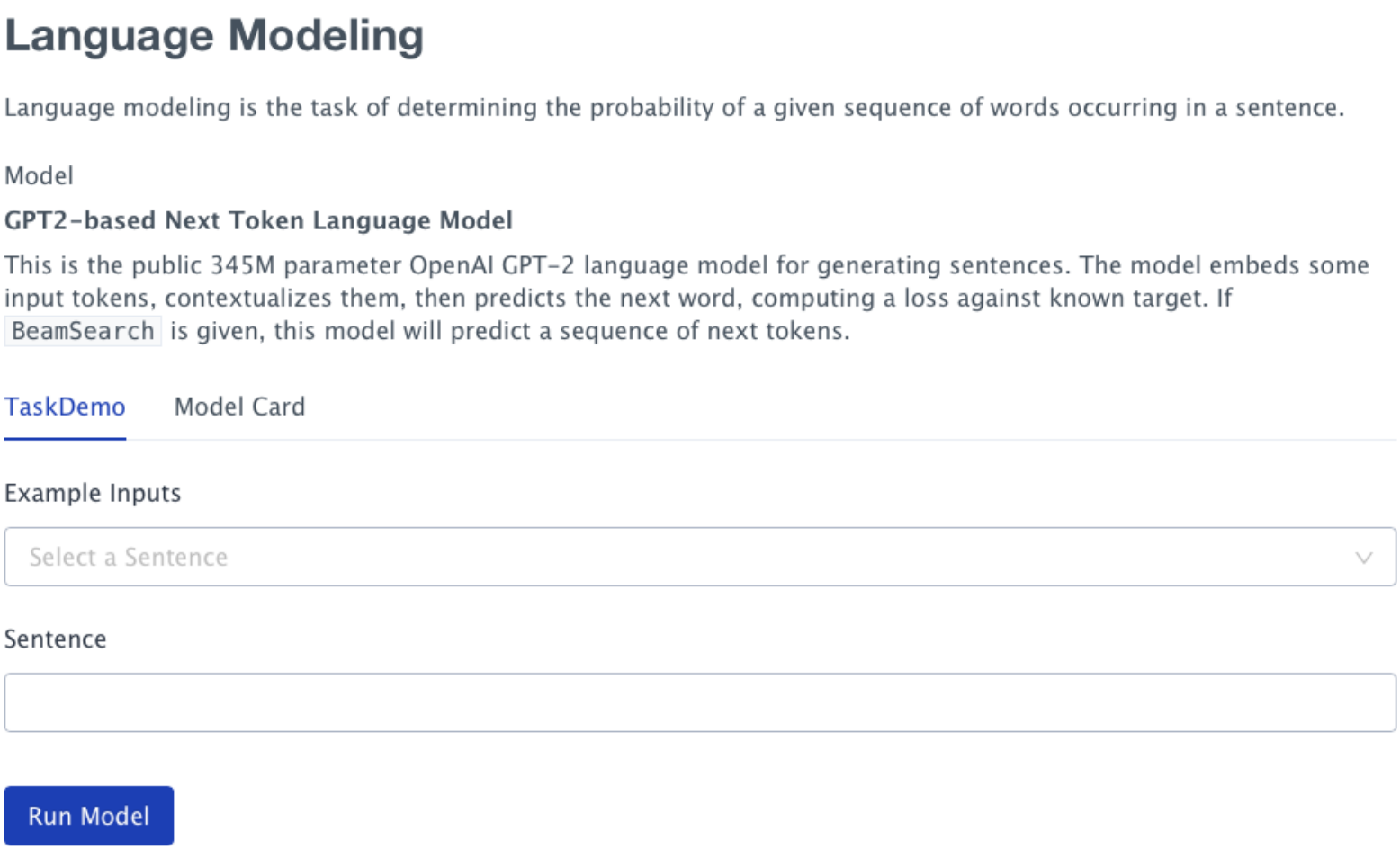
https://aclanthology.org/W19-4302/

Research papers

Title: Emergent Abilities of Large Language Models
Authors: Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama *et al.*
Repository: arXiv [cs.CL] 2022
Content: This paper discusses an unpredictable phenomenon that we refer to as emergent abilities of large language models. We consider an ability to be emergent if it is not present in smaller models but is present in larger models. The existence of such emergence implies that additional scaling could further expand the range of capabilities of language models.

https://arxiv.org/abs/2206.07682

Have fun !!



https://demo.allennlp.org/next-token-lm