Transformers

[DAT640] Information Retrieval and Text Mining

Petra Galuscakova
University of Stavanger

September 17, 2024

Recap: Natural Language Processing and Neural Networks

- Kahoot

In this module

Attention and Transformers

Contextualized word embeddings

- Static word embeddings map words with multiple senses into an average or most common-sense representation based on the training data used to compute the vectors
- The static word embedding of a word does not change with the other words used in a sentence around it (e.g. word2vec)
- In contextualized word embeddings, the representation of each token is conditioned on the context in which it is considered
- The most popular contextualized word embeddings are learned with deep neural network architecture called the Transformer architecture

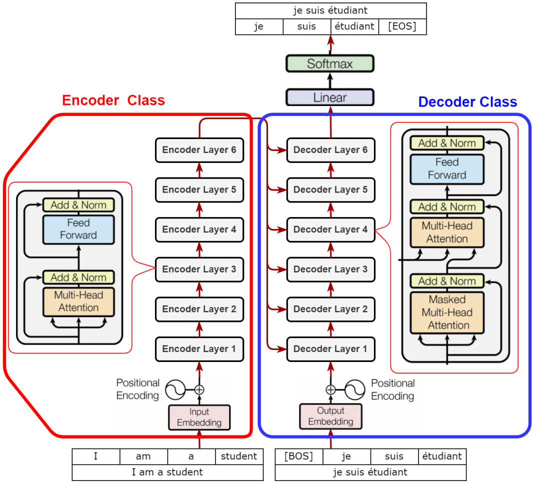
Transformers

- The transformer is a neural network designed to explicitly take into account the long-range dependencies between words
- Transformers are an example of *sequence-to-sequence* models that transform an input vectors $(x_1, ..., x_n)$ to some output vectors $(y_1, ..., y_n)$ of the same length
- Transformers are made up of stacks of transformer blocks, which are multilayer networks made by combining simple linear layers, feedforward networks, and **attention layers**
- Attention is a key innovation of the transformers which allows to directly extract and use information from arbitrarily long contexts

Transformers

- Originally, the *sequence-to-sequence* neural network is composed of two parts: an *encoder* model, which receives an input sequence and builds a contextualised representation of each input element, and a *decoder* model, which uses these contextualised representations to generate a task-specific output sequence
- Three variants made for different purposes
    - Encoder-decoder architecture (e.g., machine translation)
    - Encoder only architecture (e.g., generating contextualized embeddings)
    - Decoder only architecture (e.g., language models)

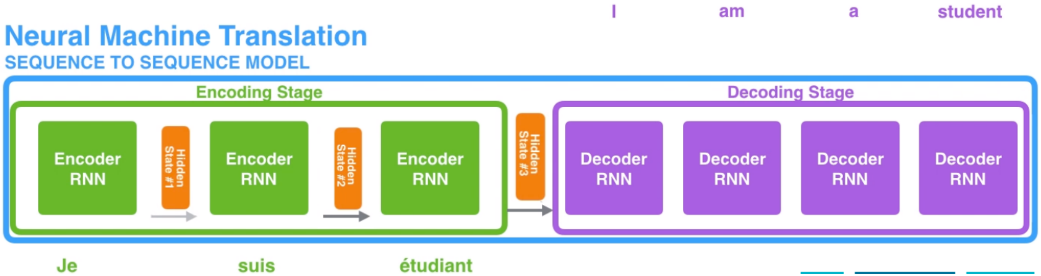
High-level architecture of Transformer model [1]



[1] Deep Learning Bible - 3. Natural Language Processing

Transformers: self-attention [2]

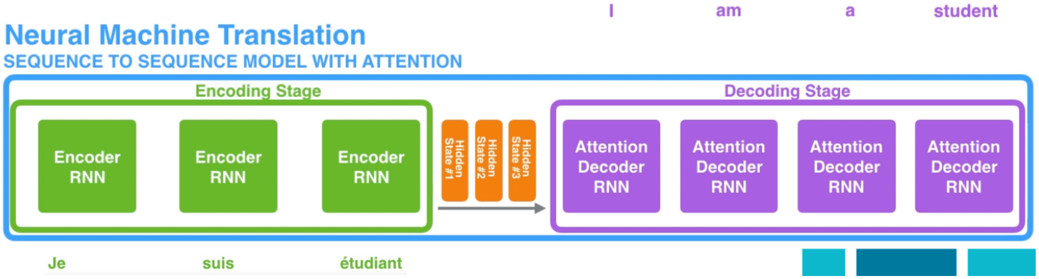
- The context vector does not allow to well process longs sentences



[2] https://jalammar.github.io

Transformers: self-attention - cont. [3]

- The attention helps to draw connections between any parts of the sequence



[3] https://jalammar.github.io

Transformers: self-attention - cont.

At the core of an attention-based approach is the ability to compare an item of interest to a collection of other items in a way that reveals their relevance in the current context. A self-attention layer allows the network to take into account the relationships among different elements in the same input.

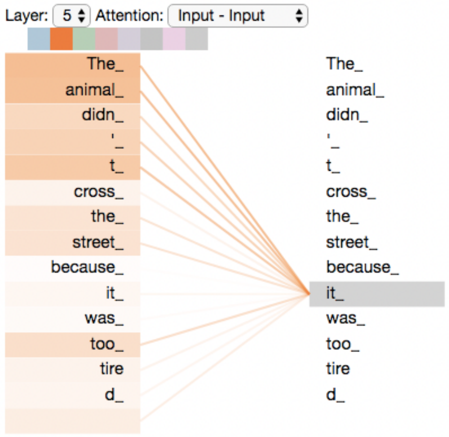


Figure: Self-attention at high level [a]

[a] https://jalammar.github.io/

Transformers: self-attention - cont.

- A self-attention layer maps input sequences $(x_1, ..., x_n)$ to output sequences of the same length $(y_1, ..., y_n)$
- Self attention differs in encoder and decoder modules
- The transformers in the *encoder* employ bidirectional self-attention layers on the input sequence or the output sequence of the previous transformer
- The transformers in the decoder employ causal self-attention on the previous decoder transformer's output and bidirectional cross-attention of the output of the final encoder transformer's output
    - In encoder module, the model has access to all of the inputs
    - In decoder module, the model has access to all of the inputs up to and including the one under consideration, but no access to information about inputs beyond the current one
- In addition, the computation performed for each item is independent of all the other computations $\rightarrow$we can easily parallelize both forward inference and training of such models

Transformers: self-attention - Implementation

- Input embedding can play three different roles during the course of the attention process:
  - The query determines the relevancy of other words in the sequence, from the perspective of this word.
  - The key determines the relevancy of this word from the perspective of other words in the sequence.
  - The value determines how salient a word is compared to the other words. We can think of a salient one as one that's been highlighted. It stands out and is important. The Transformer pays more “attention” to it.
- To capture these three different roles, transformers introduce weight matrices $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V$. These weights will be used to project each input vector $\mathbf{x}_i$ into a representation of its role as a key, query, or value

$$\mathbf{q}_i = \mathbf{W}^Q \cdot \mathbf{x}_i; \mathbf{k}_i = \mathbf{W}^K \cdot \mathbf{x}_i; \mathbf{v}_i = \mathbf{W}^V \cdot \mathbf{x}_i$$

Transformers: self-attention - Implementation[4]

Given these projections, the score between the current focus of attention, $\mathbf{x}_i$ and an element in the context, $\mathbf{x}_j$ consists of a dot product between its query vector $\mathbf{q}_i$ and the context element's key vectors $\mathbf{k}_j$:
$score(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{q}_i \cdot \mathbf{k}_j$
The output calculation for $\mathbf{y}_i$ is now based on a weighted sum over the value vectors v

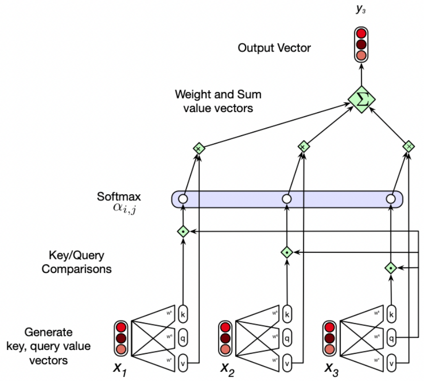


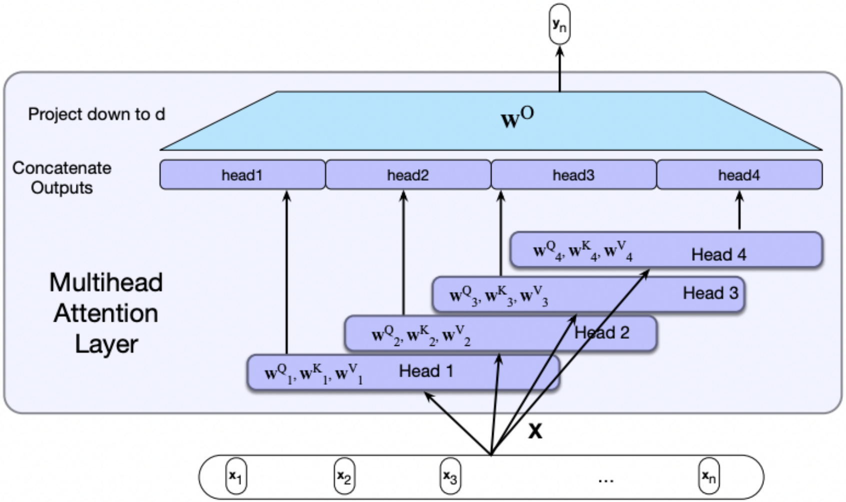
Figure: Self-attention when considering the input $\mathbf{x}_3$

[4] https://arxiv.org/pdf/2207.13443.pdf

Transformers: multihead self-attention

- Different words in a sentence can relate to each other in many different ways simultaneously
- It would be difficult for a single transformer block to learn to capture all of the different kinds of parallel relations among its inputs
- Transformers address this issue with *multihead self-attention layers*
- *Multihead self-attention layers* are sets of self-attention layers, called heads, that reside in parallel layers at the same depth in a model, each with its own set of parameters
- Each of the multihead self-attention layers is provided with its own set of key, query and value weight matrices

Transformers: multihead self-attention[5]

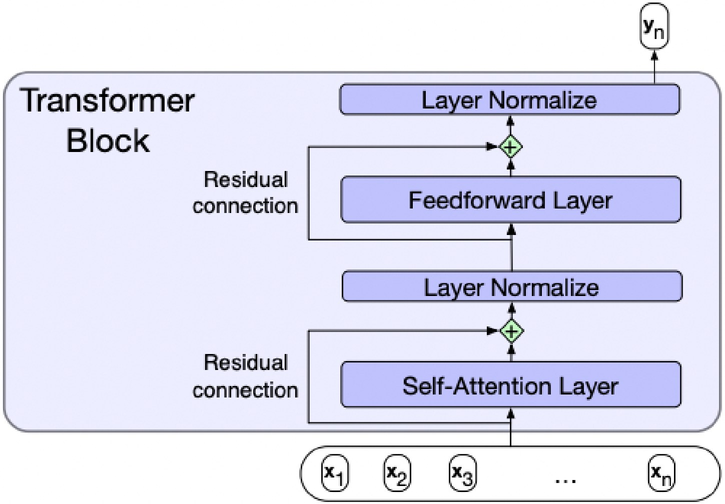


[5]`https://web.stanford.edu/~jurafsky/slp3/`

Transformers: transformer block

- The self-attention calculation lies at the core of what's called a *transformer block*, which, in addition to the self-attention layer, includes additional feedforward layers, residual connections, and normalizing layers
- A standard *transformer block* consists of a single attention layer followed by a fully-connected feedforward layer with residual connections and layer normalizations following each

Transformers: transformer block[6]



[6]https://web.stanford.edu/~jurafsky/slp3

Transformers: transformer block - cont.

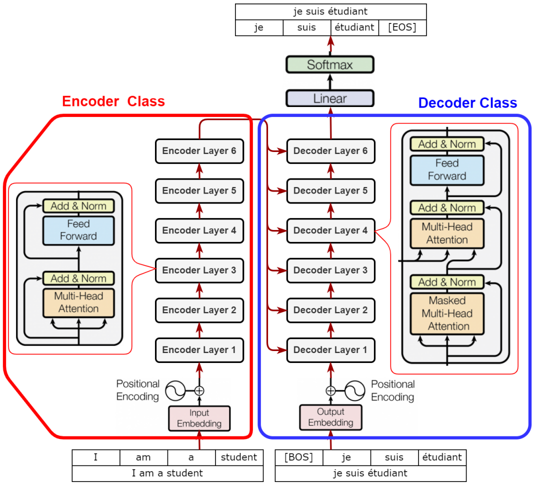
- In deep networks, residual connections are connections that pass information from a lower layer to a higher layer without going through the intermediate layer
- Allowing information from the activation going forward and the gradient going backwards to skip a layer improves learning and gives higher level layers direct access to information from lower layers
- Residual connections in transformers are implemented by adding a layer's input vector to its output vector before passing it forward
- *Layer normalization* is used to improve training performance in deep neural networks by keeping the values of a hidden layer in a range that facilitates gradient-based training

Exercise

E7-1 Self-attention in BERT-viz

Positional Embeddings

High-level architecture of Transformer model [7]

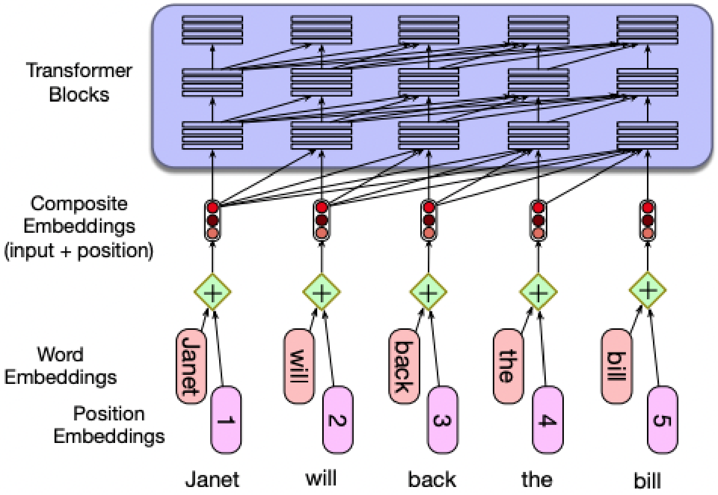


[7] Deep Learning Bible - 3. Natural Language Processing

Word embeddings

- Common tokenization algorithms are are WordPiece developed by Google to pretrain BERT or a Byte pair encoding
- The vocabulary V of the Wordpiece tokeniser is composed of 30,522 terms, where the uncommon/rare words, e.g., *goldfish*, are splitted up in sub-words, e.g., *gold##* and *##fish*
- It is trained: the vocabulary is initialized with individual characters in the language, then the most frequent combinations of symbols in the vocabulary are iteratively added to the vocabulary.

Transformers: positional embeddings[8]



[8]`https://web.stanford.edu/~jurafsky/slp3`

Transformers: positional embeddings

- Transformers modify the input embeddings by combining them with *positional embeddings* specific to each position in an input sequence
- Different ways to model the position:
  - leveraging sine and cosine functions, assign unique values to different positions in the sequence
  - learn the positional embeddings along with other parameters during training
- To produce an input embedding that captures positional information, we just add the word embedding for each input to its corresponding positional embedding

Transformer Architectures

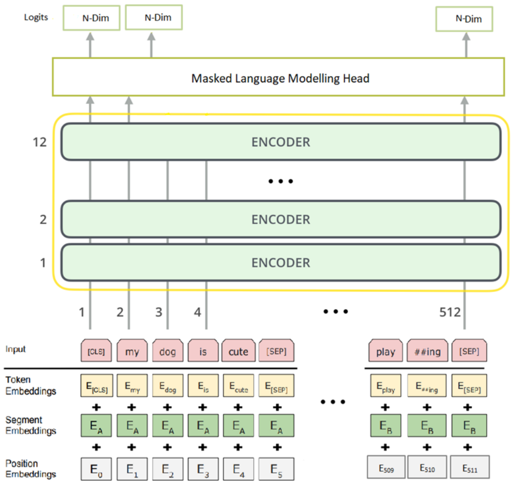
Transformer Architectures

- Encoder (e.g. BERT)
- Encoder-decoder (e.g. T5)
- Decoder (e.g. GPT)

Transformers: encoder-only models

- *Encoder-only* models receive as input all the tokens of a given input sentence, and they compute an output contextualised word embedding for each token in the input sentence
- Representatives of this family of models include BERT, RoBERTa, and DistilBERT
- Encoder-only models are trained as *masked language models (MLM)*
- MLM training focuses on learning to predict missing tokens in a sequence given the surrounding tokens

BERT model[9]

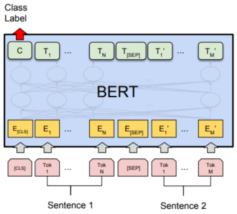


[9]https://www.analyticsvidhya.com/blog/2022/09/fine-tuning-bert-with-masked-language-modeling/

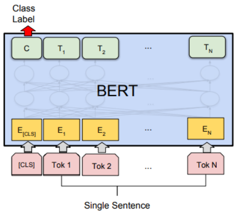
BERT model - cont.

- BERT input text is tokenised using the WordPiece sub-word tokeniser
- The first input token of BERT is always the special *[CLS]* token, that stands for "classification"
- BERT accepts as input other special tokens, such as *[SEP]*, that denotes the end of a text provided as input or to separate two different texts provided as a single input
- BERT accepts as input at most 512 tokens, and produces an output embedding in $\mathbb{R}^l$ for each input token
- The most commonly adopted BERT version is *BERT-base*, which stacks 12 transformer layers, and whose output representation space has $l = 768$ dimensions

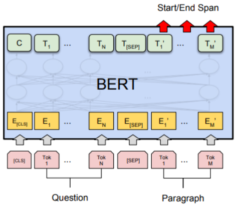
BERT model - applications



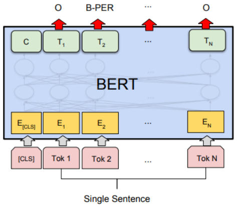
(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG



(b) Single Sentence Classification Tasks:
SST-2, CoLA



(c) Question Answering Tasks:
SQuAD v1.1



(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

BERT model - query/document similarity

- Given a query-document pair, both texts are tokenised into token sequences $q_1, ..., q_m$ and $d_1, ..., d_n$
- Then, the tokens are concatenated with BERT special tokens to form the following input configuration that will be used as BERT input:

$$[CLS]q_1, ..., q_m[SEP]d_1, ..., d_n[SEP]$$

- The self-attention layers in the BERT encoders are able to take into account the semantic interactions among the query tokens and the document tokens
  - The attention mechanism in encoder models is bi-directional
- The output embedding $\mu_{[CLS]} \in \mathbb{R}^l$, corresponding to the input *[CLS]* token, serves as a contextual representation of the query-document pair as a whole

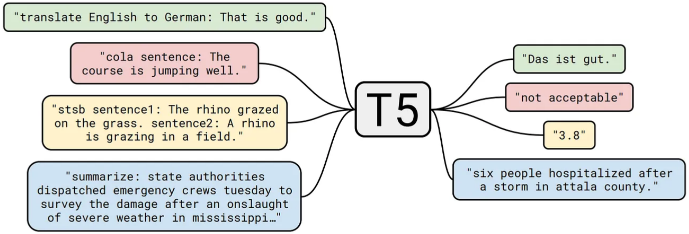
Encoder-decoder models

Transformers: encoder-decoder models

- In the *encoder-decoder* model, the encoder model receives as input all the tokens of a given sequence and builds a contextualised representation, and the decoder model sequentially accesses these embeddings to generate new output tokens, one token at a time
- Representatives of this family of models include BART and T5

Encoder-decoder models - training[10]

- Encoder-decoder models are trained as *casual language models (CLM)*
- CLM training focuses on predicting the next token in an output sequence given the preceding tokens in the input sequence
- Prompting helps guide language model behavior by adding some input text specific to a task.


"translate English to German: That is good."
"cola sentence: The course is jumping well."
"stsb sentence1: The rhino grazed on the grass. sentence2: A rhino is grazing in a field."
"summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of severe weather in mississippi…"
T5
"Das ist gut."
"not acceptable"
"3.8"
"six people hospitalized after a storm in attala county."

[10] https://blog.research.google/

Encoder-decoder models - prompt learning

- In *prompt learning* the input texts are reshaped as a natural language template, and the downstream task is reshaped as a cloze (fill in the blanks) task
- For example, in topic classification, assuming we need to classify the sentence *text* into two classes c_0 and c_1, the input template can be: $Input : text\ Class : [OUT]$
- Among the vocabulary terms, two label terms w_0 and w_1 are selected to correspond to the classes c_0 and c_1, respectively
- The probability to assign the input text to a class can be transferred into the probability that the input token *[OUT]* is assigned to the corresponding label token:

$$p(c_0|text) = p([OUT] = w_0|Input : text\ Class : [OUT])$$

$$p(c_1|text) = p([OUT] = w_1|Input : text\ Class : [OUT])$$

Fine-tuning T5 model for computing relevance score

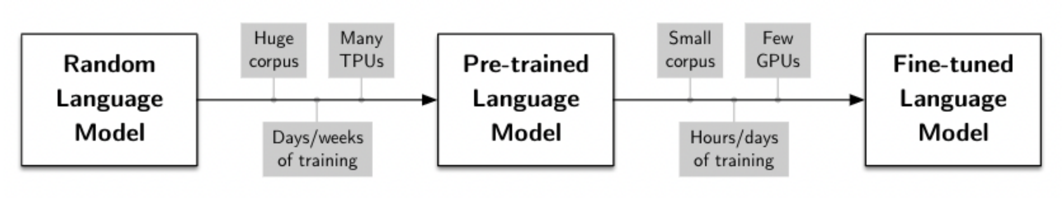
- In prompt learning approach for relevance ranking using a T5 model, the query and the document texts q and d are concatenated to form the following input template:

$$Query : q\ Document : d\ Relevant : [OUT]$$

- An encoder-decoder model is fine-tuned with a downstream task taking as input this input configuration, and generating an output sequence whose last token is equal to *True* or *False*, depending on whether the document d is relevant or non-relevant to the query q
- The query-document relevance score is computed by normalising only the *False* and *True* output probabilities, computed over the whole vocabulary, with a *softmax* operation

Pre-training and Fine-tuning

Transfer learning[11]



[11] https://arxiv.org/pdf/2207.13443.pdf

Pre-trained Language Models

- Transformer-based models are usually trained using massive text data to obtain *pre-trained language models*
- It allows the model to learn general-purpose knowledge about a language that can be adapted afterwards to a more specific *downstream task*
- This approach is called *transfer learning*
- In *transfer learning*, a pre-trained language model is used as initial model to *fine-tune* it on a domain-specific, smaller training dataset for the downstream target task
- *Fine-tuning* is the procedure to update the parameters of a pre-trained language model for the domain data and target task

Fine-tuning for IR

- The pre-trained language models adopted in IR require a fine-tuning on a specific downstream task
- Given an input query-document pair *(q,d)* a neural IR model $M(\theta)$, parametrised by θ, computes $s_\theta(q,d) \in \mathbb{R}$, the score of the document *d* w.r.t. the query *q*
- We want to predict $y \in \{+,-\}$
- We sample correct and incorrect pairs
- Training objective: *the score function must assign a high score to a relevant document and a low score to a non-relevant document*

Fine-tuning for IR - cont.

- Typically, a dataset T available for fine-tuning pre-trained language models for relevance scoring is composed of a list of triples (q, d^+, d^-), where q is a query, d^+ is a relevant document for the query, and d^- is a non-relevant document for the query
- In this case, the expected cross entropy is approximated by the sum of the cross entropies computed for each triple:

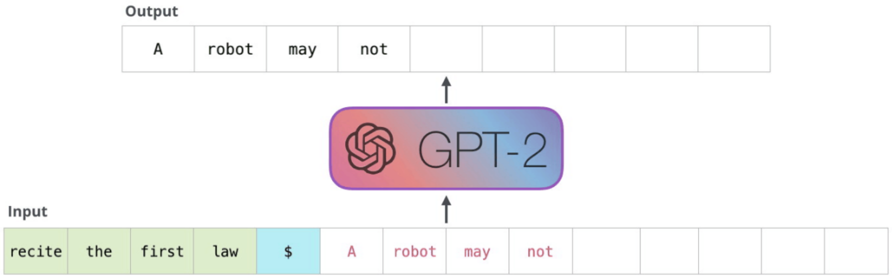
$$\mathbb{E}[l_{CE}(y,(q,d))] \approx \frac{1}{2|T|} \sum_{(q,d^+,d^-)\in T} (-log(s_\theta(q,d^+)) - log(1 - s_\theta(q,d^-)))$$

Large Language Models (LLMs)

Decoder-only models

- Decoder models use only the decoder part of a Transformer model
- At each stage, for a given word the attention layers can only access the words positioned before it in the sentence
- These models are best suited for tasks involving text generation
- These models are often called auto-regressive models (they assume that the current value of a time series is a function of its past values)
- An example of are GPT, Gemini, BLOOM, LLaMA models

Autoregressive models[12]

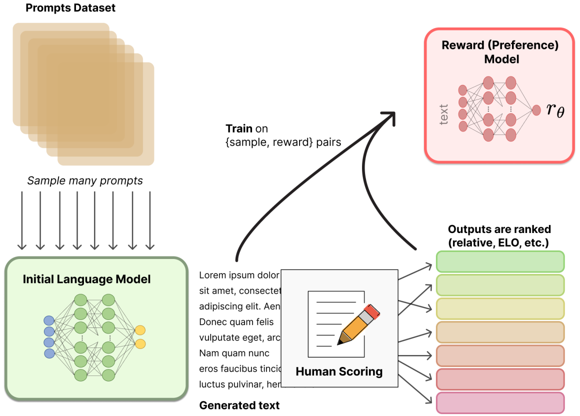


[12] https://medium.com/@YanAIx/step-by-step-into-gpt-70bc4a5d8714

Generative Model pre-Training

- The training of decoder models usually revolves around predicting the next word in the sentence, i.e. the probability $p(x_i x_{1:i-1})$ in unsupervised way
- Afterwards, the model might be fine-tuned using reinforcement learning with human feedback to produce accurate and realistic results

LLM training - reinforcement model[13]



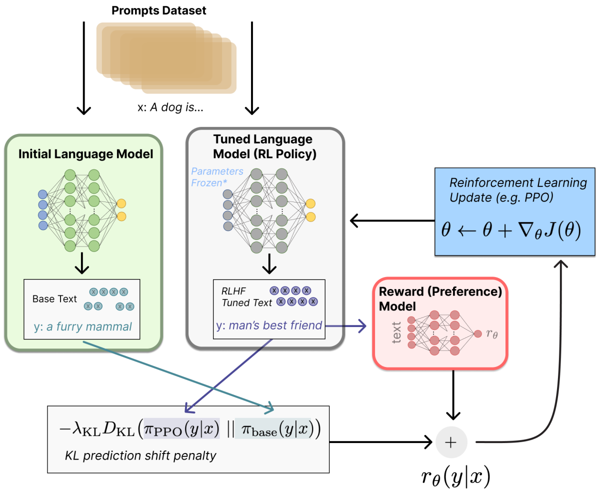
Prompts Dataset
Sample many prompts
Initial Language Model
Lorem ipsum dolor sit amet, consectet adipiscing elit. Aen Donec quam felis vulputate eget, arc Nam quam nunc eros faucibus tincid luctus pulvinar, her
Generated text
Human Scoring
Train on {sample, reward} pairs
Reward (Preference) Model
text
r_θ
Outputs are ranked (relative, ELO, etc.)

[13] https://huggingface.co/blog/rlhf

LLM training - reinforcement model - cont.

- Successfully used in ChatGPT
- With a language model, one needs to generate data to train a reward model (RM), which is how human preferences are integrated into the system
- The training dataset of prompt-generation pairs for the RM is generated by sampling a set of prompts from a predefined dataset
- The prompts are passed through the initial language model to generate new text
- Human annotators are used to rank the generated text outputs from the LM
- Next, reinforcement learning (RL) is used to optimize the original language model with respect to the reward model

LLM training - reinforcement learning[14]


Prompts Dataset
x: A dog is...
Initial Language Model
Base Text
y: a furry mammal
Tuned Language Model (RL Policy)
Parameters Frozen*
RLHF Tuned Text
y: man's best friend
Reinforcement Learning Update (e.g. PPO)
$\theta \leftarrow \theta + \nabla_\theta J(\theta)$
Reward (Preference) Model
text
r_θ
$-\lambda_{\text{KL}} D_{\text{KL}}(\pi_{\text{PPO}}(y|x) \,||\, \pi_{\text{base}}(y|x))$
KL prediction shift penalty
+
$r_\theta(y|x)$

[14] https://huggingface.co/blog/rlhf

Supervised fine-tuning

- Fine-tuning involves updating the weights of a pre-trained model by training on a supervised dataset specific to the desired task
- Typically thousands to hundreds of thousands of labeled examples are used
- The main advantage of fine-tuning is strong performance on many benchmarks
- The main disadvantages are the need for a new large dataset for every task, the potential for poor generalization
- Though it is possible to fine-tune GPT models, they do not require fine-tuning

LLM tuning - few shot learning

- The model is given a few demonstrations of the task at inference time as conditioning, but no weight updates are allowed
- The main advantages of few-shot are a major reduction in the need for task-specific data and reduced potential to learn an overly narrow distribution from a large but narrow fine-tuning dataset.
- A small amount of task specific data is still required
- sea otter => loutre de mer
  peppermint => menthe poivrée
  plush girafe => girafe peluche
  cheese =>

LLM tuning - zero shot

- When no demonstrations are used, and the model is only given a natural language instruction describing the task, then the approach is zero shot
- It is closest to how humans perform tasks, it provides maximum convenience, but is also the most challenging
- Translate to French: cheese $=>$

LLM tuning - one shot

- It is the same as few-shot except that only one demonstration is allowed
- The reason to distinguish one-shot from few-shot and zero-shot is that it most closely matches the way in which some tasks are communicated to humans
- Also, it is sometimes difficult to communicate the content or format of a task if no examples are given

Exercise

E7-2 - ChatGPT Exercise

Models parameters[15]

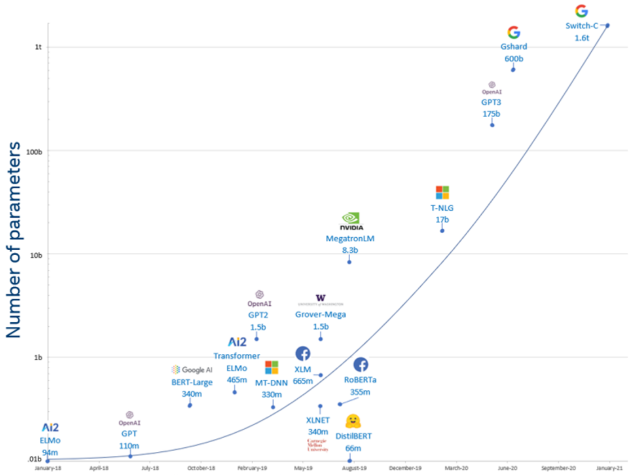
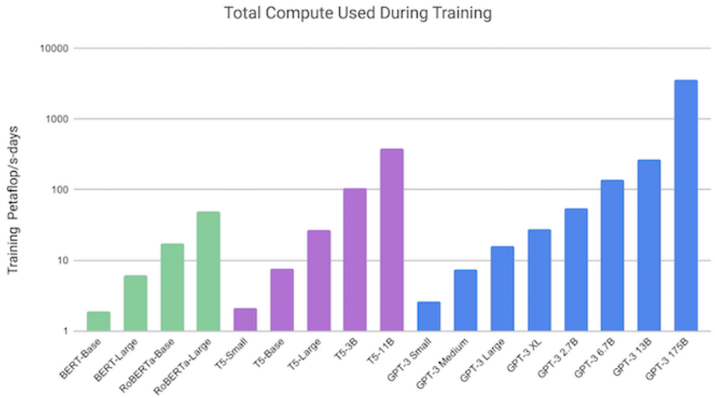


Figure 1: Exponential growth of number of parameters in DL models

[15] https://heits.digital/articles/gpt3-overview

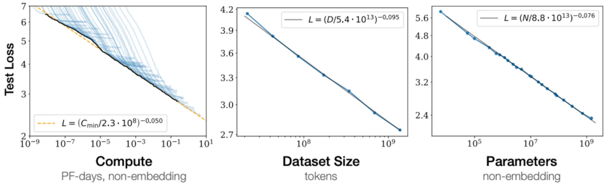
List of LLMs[16]



[16] `https://magazine.sebastianraschka.com/p/understanding-encoder-and-decoder`

Scaling Laws

- The scaling laws of LLMs shed light on how a model's quality evolves with increases in its size, training data volume, and computational resources.
- Two significant findings:
  - The impact of scale (number of parameters (N), the size of the dataset (D), and the computational resource) on the loss of models is more pronounced than the influence of the model's architectural structure.
  - There is a power-law relationship between the performance of the model and each of the scaling factors (N, D, C) when they are not constrained by one another



Demo

LM Arena: https://lmarena.ai/

Summary

- Attention and transformers
- Transformer architectures
- LLMs

Reading and References

- *Speech and Language Processing, Chapter 9 (Transformers) and 10 (Large Language Models)*, Dan Jurafsky and James H. Martin [17]
- *The Illustrated Transformer*, Jay Alammar [18]

[17] `https://web.stanford.edu/~jurafsky/slp3/`
[18] `https://jalammar.github.io/illustrated-transformer/`