Essence of LLMs and Transformers I

Vaibhav Kalvakota

Bayesian Group

June, 2024

Based on upcoming mini-paper on sparsity Transformers

Related notes will be posted on my website
vkalvakotamath.github.io/files/2024-LLM-1.

Natural Language Processing is very important.

Large Language Models are a subset of NLP.

"Why are polar bears cute?" Relevant tokens: polar bears (animal), cute (characteristic), why (question, output has to be an answer).

"Polar bears are often seen as "cute" because of a combination of physical traits and behaviors that we humans tend to find endearing. Some of the reasons include:
Fluffy Appearance: Polar bears have thick fur that looks soft and fluffy, giving them a cuddly appearance, especially in pictures or videos. Their fur covers their entire body, including their round ears and large paws, enhancing the sense of softness..."

Very large chunks of data and parameters to be trained. Typically a three stage process:

$$\textbf{Pretraining} \longrightarrow \textbf{Fine-tuning} \longrightarrow \textbf{Repeat!} \tag{1}$$

To work with the LLM, we essentially want to work out these in large chunks of data. After we obtain a base model, we fine-tune parameters and try to make the model "better"

LLMs, fundamentally: Suppose you had a sequence of N words and wanted to train a model to predict the next word. That is, you want to find

$$p(x_N|\mathbf{X}_{N-1}) \,. \tag{2}$$

That is, given $N-1$ *context* words $\mathbf{X}_{N-1} = \{x_1 \dots x_{N-1}\}$, what is the $p(x_N)$ given those inputs?

We want to somehow (1) keep a track of those $N-1$ inputs, (2) "encode" positional information ("how did the cat eat the cheese" is different from "how did the cheese eat the cat") and (3) optimize this model in all sorts of ways.

The how: Encode positional information, add contextualization, use *neural networks*, and obtain a base model.

This is not as easy as it sounds, lmao.

Autoregressive Generative models

Objective: Make an NLP model that can understand meaningful contextualization of inputs in a sequence $\Sigma(x_1, x_2 \ldots x_n)$ and generate a meaningful output.

How do you do this?

Early example was of **Recurrent neural networks** (RNNs) that would store a state Ψ based on input before processing the next input.

That is, let $x_i(t_i)$ be the input at $t = i$. Then, we store in a "hidden layer" a state $\Psi(t = i)$, which will be passed on to input $x_{i+1}(t_{i+1})$.

$$
\begin{tikzcd}
x_t \arrow[r] \arrow[d] & HL - \Psi_t \arrow[d] \arrow[dr] & \\
O_t & x_{t+1} \arrow[r] \arrow[d] & HL - \Psi_{t+1} \arrow[d] \\
 & O_{t+1} & x_{t+2}
\end{tikzcd}
$$

This is a vanilla forward propagating RNN with one forward pass hidden layer.

There are also bidirectional RNNs where there is a forward and a backward propagating hidden layer.

$$\begin{tikzcd}
x_{t-1} \arrow[rrr] \arrow[dr] & & & HL_F - \Psi_{t-1} \arrow[dl] \arrow[dd] \\
& HL_B - \Psi_{t-1} \arrow[r] \arrow[u] & O_{t-1} & \\
x_t \arrow[rrr] \arrow[dr] & & & HL_F - \Psi_t \arrow[dl] \arrow[dd] \\
& HL_B - \Psi_t \arrow[r] \arrow[uu] & O_t & \\
x_{t+1} \arrow[rrr] \arrow[dr] & & & HL_F - \Psi_{t+1} \arrow[dl] \arrow[dd] \\
& HL_B - \Psi_{t+1} \cdots \arrow[r] \arrow[uu] & O_{t+1} & \\
& & &
\end{tikzcd}$$

Then there are also LSTM models which are VERY good.

They don't have the vanishing gradient problem, and are significantly better at long sequence contextualization.

However, the breakthrough came with **Attention Is All You Need** by **Vaswani et al, 2017**.

Attention Is All You Need

Ashish Vaswani[*]
Google Brain
avaswani@google.com

Noam Shazeer[*]
Google Brain
noam@google.com

Niki Parmar[*]
Google Research
nikip@google.com

Jakob Uszkoreit[*]
Google Research
usz@google.com

Llion Jones[*]
Google Research
llion@google.com

Aidan N. Gomez[* †]
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser[*]
Google Brain
lukaszkaiser@google.com

Illia Polosukhin[* ‡]
illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

[*] Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

[†] Work performed while at Google Brain.

[‡] Work performed while at Google Research.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

Wait!

ML models are computationally expensive!

Base **LLMs** use fairly short code on nearly 50TB of text data, 500GB of parameters, MANY GPUs for the duration of training, very costly both computationally as well as financially!

Model architecture optimization is a real dealbreaker. Bad optimization and neural network architecture = bad models = BAD!

Transformers: Tokenize, positionally encode, self-attention, MLPs, etc. etc. Optimization involves making the model more **efficient**. This means making the model *faster*, *productive* and optimization over parameters.

But we are at a very good place! Transformers > RNNs and LSTMs. Computationally efficient and not bulky to work with!

Base **GPT**, **Claude** and **Llama**.

The How

Obtain a **base model** after pretraining.

Soft-launch for testing; next step is **fine-tuning**.

Retrain all θ's with autoregression. In some cases, use PEFT or parameter-efficient fine-tuning to retrain only select parameters.

Use gradient descent algorithm to minimize negative log-likelihood $\rightarrow$ minimize cross-entropy loss J_{CE}.

Fine-tune θ_N to find the optimal parameters for the model.

Use RLHF to maximize reward by human feedback.

Obtain a better model, go back to step 1 with new and better optimized base model.

The How

Init: causal LMs with self-attention, where

$$\mathrm{att}_t = \sum_{t \geq T} \phi_{tT} \mathbf{X}_T \ . \tag{3}$$

That is, input attention correlations are *only* forward propagating.

Problem: Backward contextualization does not happen.

$$\text{I absolutely \_\_\_\_\_ Dune Part 2 .} \tag{4}$$

Since context is forward propagating, it would not predict **love** in the blank.

Solution: Bidirectional LMs.

Attention correlations propagate *throughout* the entire input sequence.

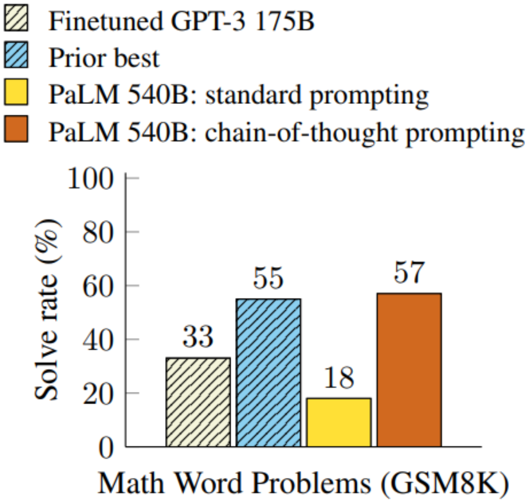
CoT

Improving the performance of LLMs on evaluation benchmarks is very important. As of today (28th of September, 2024 0600 hrs New York time), two of the best performing LLMs are GPT-4o and Claude 3.5 Sonnet, with 3.5 Sonnet performing better in GPQA and HumanEval and GPT-4o in MMLU and mathematical evaluation.

In these benchmarks one of the many forerunners are CoT or Chain of Thought LLMs.

As an example, to [input] *how many apples does Paul Atreides have if he starts with 12, eats 6, gives 2 to Chani and 1 to Stilgar?* [/input], a badly performing model could say [output] 5 [/output]. With CoT, it could perform better reasoning and ergo better arithmetic calculations.

To illustrate an example of why CoT improves LLMs, see this plot from **Wei et al, 2022**:


Finetuned GPT-3 175B
Prior best
PaLM 540B: standard prompting
PaLM 540B: chain-of-thought prompting
Solve rate (%)
0
20
40
60
80
100
33
55
18
57
Math Word Problems (GSM8K)

Rank* (UB)	Delta	Model	Arena Score	95% CI	Votes	Organization	License	Knowledge Cutoff
1 ↑	2	o1-mini	1373	+16/-14	1967	OpenAI	Proprietary	2023/10
1	0	o1-preview	1359	+14/-13	1825	OpenAI	Proprietary	2023/10
2 ↓	-1	ChatGPT-4o-latest (2024-09-03)	1339	+8/-11	3359	OpenAI	Proprietary	2023/10
4 ↑	3	Claude 3.5 Sonnet	1295	+6/-7	13024	Anthropic	Proprietary	2024/4
4 ↑	2	GPT-4o-2024-05-13	1294	+5/-6	19559	OpenAI	Proprietary	2023/10
4 ↑	3	Meta-Llama-3.1-405b-Instruct-bf16	1289	+12/-13	1746	Meta	Llama 3.1 Community	2023/12
4	0	Grok-2-08-13	1288	+8/-11	5539	xAI	Proprietary	2024/3
4	0	Gemini-1.5-Pro-Exp-0827	1287	+8/-8	6168	Google	Proprietary	2023/11
4 ↑	3	GPT-4o-mini-2024-07-18	1283	+8/-9	6279	OpenAI	Proprietary	2023/10
4 ↑	11	Deepseek-v2.5	1283	+14/-14	1678	DeepSeek	DeepSeek	Unknown
4 ↑	6	Qwen2.5-72b-Instruct	1278	+17/-20	1252	Alibaba	Qwen	2024/9
6 ↑	1	Meta-Llama-3.1-405b-Instruct-fp8	1277	+10/-9	6350	Meta	Llama 3.1 Community	2023/12
6 ↑	2	GPT-4o-2024-08-06	1275	+10/-9	4707	OpenAI	Proprietary	2023/10
6 ↑	10	Mistral-Large-2407	1271	+8/-9	6007	Mistral	Mistral Research	2024/7
6 ↑	23	Deepseek-Coder-v2-0724	1266	+14/-12	2170	DeepSeek	Proprietary	Unknown
10 ↑	5	GPT-4-Turbo-2024-04-09	1264	+6/-4	19304	OpenAI	Proprietary	2023/12
11 ↓	-4	Grok-2-Mini-08-13	1261	+7/-8	4851	xAI	Proprietary	2024/3
11 ↓	-4	Gemini-1.5-Flash-Exp-0827	1258	+11/-9	5264	Google	Proprietary	2023/11

Scaling Laws

How do you know what resources and how much time it will take to implement an LLM?

Not a trivial question.

Scaling laws: three primary factors to consider:

1. Amount of training data,
2. # of compute and resources (Limitations, GPUs, etc),
3. # of parameters and complexity for fine-tuning, etc.

There are other factors like the neural network model itself, vocabulary size, etc.

See **Kaplan et al, 2020** on scaling laws. Trained decoder-only Transformer on WebText2 with BPE and $N_{\text{vocab}} = 50257$, $N_{Context} = 1024$ and loss function = negative log-likelihood minimization or cross-entropy loss.

The Why

What makes Transformers better than RNNs and LSTMs?

Primarily three factors:

1. **Parallelization:** RNNs and LSTMs don't have parallelization because $HL - \Psi_{t-1}$ is needed to infer anything about x_t and O_t,
2. **RNNs are "strongly causal"**, in the sense that they depend on the last $t - n$ $(t > n)$ inputs and have vanishing gradient problems w.r.t backpropagation. This means bad long range dependencies. LSTMs correct this. Transformers do not have this issue at all.
3. **Positional encoding** makes Transformers better than RNNs and LSTMs.

	RNNs	**LSTMs**	**Transformers**
Vanishing Gradient	Significant	Not an issue	Not an issue
Long-term Dependencies	Bad	Somewhat better	Very effective
Complexity	$O(nd^2)$	$O(nd^2)$	$O(n^2d)$
Parallelization	By sequence	By sequence	Parallel computation
Memory Usage $\propto$	$O(n \cdot d)$	$O(n \cdot d)$	$O(n^2 + nd)$
Context Handling	By sequence	By sequence	Parallel

While Transformer complexities and memory usages are high, they can use GPUs and TPUs VERY effectively to parallel compute matrix multiplications and work very well for LLMs.

There are ways to optimize Transformers.

Two things I am interested in: (1) **knowledge distillation**, and (2) **adaptive/learnable sparse models**.

Fascinating links to scaling laws, which are yet to be clear.

Vanilla Transformer models compute attention weights with $\mathrm{softmax}$. Since $\mathrm{softmax}(z) \propto \exp(z)$, this value can never be 0. This refers to the **dense** nature of attention weights in the model.

This also implies that there are irrelevant-relevant input correlations in the model that cannot be minimized. Artificial clipping might help, but not clear how to do this right.

Adaptive sparsity by **Correia et al, 2019** tries to change this by replacing $\mathrm{softmax}$ with $\alpha - \mathrm{entmax}$. Superficially, this allows the model to be **sparse**.

By treating α as a learnable dynamic parameter, it is possible to make the model more accurate.

Makes complexity go down from $O(n^2)$ as previously seen. But how would they change performance-wise with GPUs and TPUs?

Open problems: Scaling laws, dynamics of complexity-accuracy trade-offs, etc?

The Sus

AI safety is important!

Like e/acc, but at what cost? Better to ensure ethical and moral grounds for AI are concrete.

E.g. don't let jack-a's ask your LLM how to make a malware, or illegal web scraping.

Prompt injections, unintentional unethical data in training, etc. have to be considered. Example: PDF/text reading LLMs could could go through a corrupt PDF/doc with hidden/disguised prompt injection code and send GET requests to attacker servers with info in the url. This is bad!

This doesn't happen with high-level LLMs.

The Sus

This doesn't have to be so high-level either. Corrupt or malformed training data is sufficient to make LLMs make bad predictions or produce misinformation.

From a code development POV, since coding with AI is very popular, it is important to ensure LLM-generated code lies within ethics and copyrights.

E.g. Google Colab with Gemini-produced code shows citations for code suggestions. Very good!

Arguably, in text-to-art generative models these ethics become more stronger and difficult to base on.

Would generative models be reiterating and generating based on someone's art design and would this count, technically at least, as plagiarism?

Finally...

LLMs are awesome.

You can make a GPT-like LLM yourself! assuming you have the money, compute and time for it.

There are many interesting problems and factors to work with.
And ethical principles to stick to.

Not formal, but it is worth to work with LLMs!

Thank you for your **attention**!