https://medium.com/@yumo-bai/why-are-most-llms-decoder-only-590c903e4789

Why are most LLMs decoder-only?

Dive into the rabbit hole of recent advancement in Large Language Models

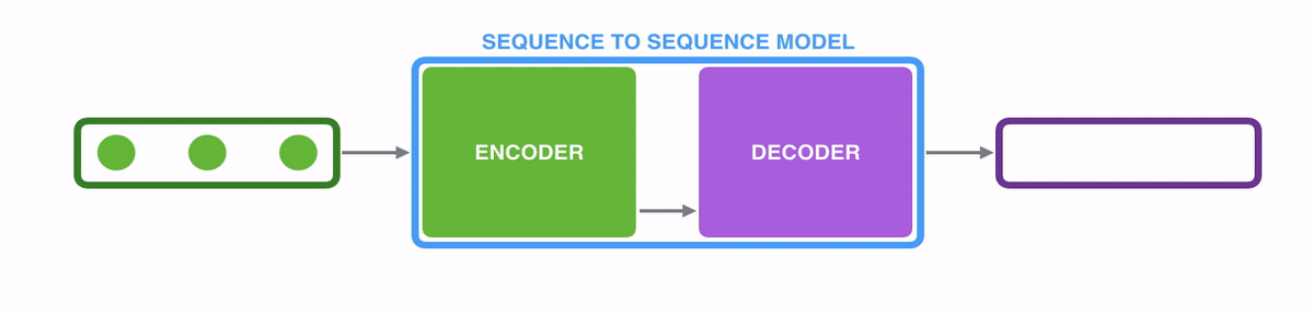
I came across this question during mentoring for DeltaHacks and could not come up with an answer that was persuasive enough for myself. So I did some digging and it turned out to be a fascinating rabbit hole to dive down into. It involves a blend of understanding transformers, architectures, mathematics, and engineering optimizations. Here I will share the two cents that I picked up along the way.

Overview of Language Model Architectures

Let's first familiarize ourselves with some architectural terms.

Encoder and Decoder

- Encoder: Processes and transforms input data into a condensed representation, capturing the essential information. In a translation task, an encoder takes an English sentence and converts it into a vector that represents its linguistic features and meaning.
- Decoder: Takes the encoded representation and generates an output, often in a different form. In the same translation task, the decoder takes the encoded representation of a sentence in English and generates its equivalent in French.



Source: Visualizing A Neural Machine Translation Model (Great post by Jay Alammar)

Encoder-Only Models

- Example: BERT-based models
- Pretraining Approach: Masked Language Modelling (MLM)
- Use Case: Tasks that require a deep understanding of input data. These models are effective for classification, sentiment analysis, and information extraction.

Decoder-Only Models

- Example: GPT, XLNet
- Pretraining Approach: Next Token Prediction
- Use Case: Generative tasks. They work by predicting subsequent text based on the provided context in an auto-regressive fashion. Their primary function is output generation without a separate encoding phase.

Encoder-Decoder Models

- Example: T5, BART, Google Gemini (Probably)
- Pretraining: Task-dependent
- Use Case: Tasks that involve both understanding and generating data. They first encode an input sequence into an internal representation and then decode this representation into an output sequence.

Comparing the purpose of these architectures, we can first easily exclude encoder-only models: They are typically pre-trained with MLM and do not necessarily help with generating output.

The decoder-only ones, on the other hand, make perfect sense: They are used to generate outputs and are pre-trained on Next Token Prediction tasks, which is exactly the task for most LLMs.

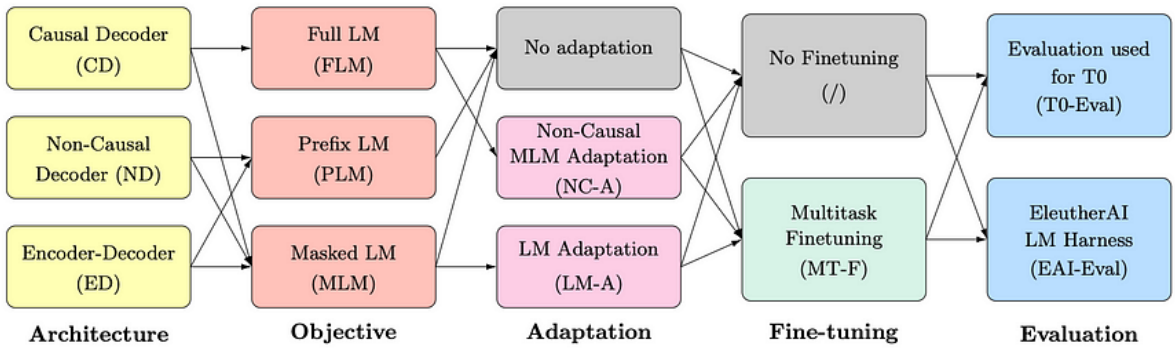
The question really boils down to Decoder-only versus Encoder-Decoder architecture:

> Having a decoder component and thereby generative ability, wouldn't having the extra encoder components only help?

Causal Decoder (CD) vs Encoder-Decoder (ED)

The performance of decoder-only, also referred to as causal decoder, against encoder-decoder models has long been studied. One of the earlier works is the paper What Language Model Architecture and Pretraining Objective Work Best for Zero-Shot Generalization? by Wang et al, published on ICML 2022. In this study, researchers compared various combinations of architecture and pretraining approaches. What they found is that:

> Our experiments show that causal decoder-only models trained on an autoregressive language modeling objective exhibit the strongest zero-shot generalization after purely self-supervised pretraining.
>
> However, models with non-causal visibility on their input trained with a masked language modeling objective followed by multitask finetuning perform the best among our experiments.



Combinations studied in the ICML paper

Ok great, so encoder-decoder > decoders-only > encoder-only, right?

Well turns out that even though the aforementioned paper revealed some valuable insights into developing larger models. Some other factors need to be considered when choosing the architecture.

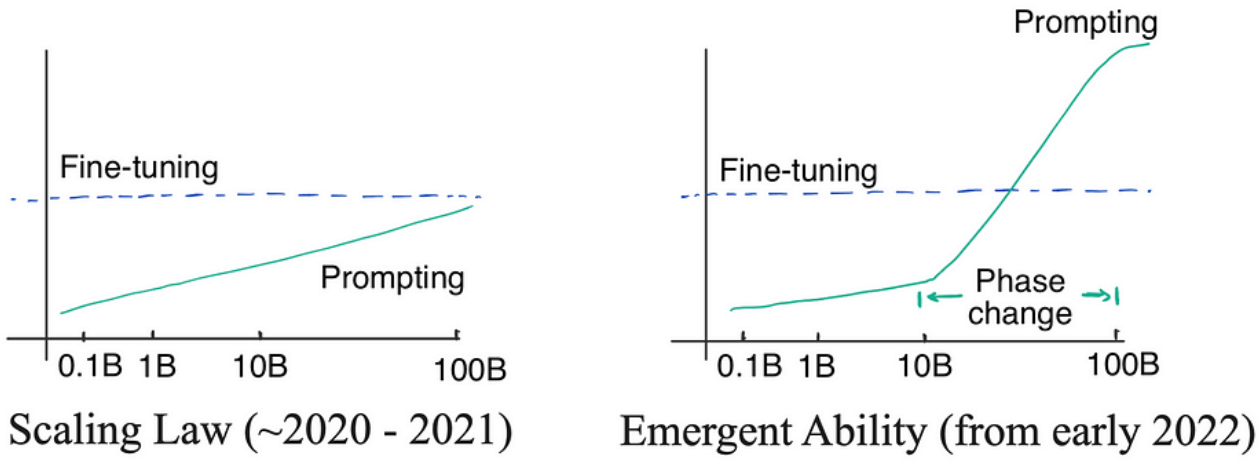
Cost of Training

To achieve the maximum potential of ED, we would need to perform multitask finetuning (which is basically instruction finetuning) on labeled data and it could be very expensive, especially for the larger models.

CD models on the other hand achieve great performance for their strong zero-shot generalization which works nicely with the current convention — self-supervised learning on a large-scale corpus.

Emergent Ability

The models compared in the paper have around 5B parameters and are trained over 170B tokens. It's not large enough to account for something miraculous — *the emergent ability* of LLMs.



Source: A Closer Look at Large Language Models Emergent Abilities by Yao Fu

Emergent abilities in Large Language Models (LLMs) refer to the phenomenon where models display new, sophisticated capabilities not explicitly taught during training, arising naturally as the model scales in size and complexity. Here is a great blog on emergent abilities by Yao Fu where you can learn more about it.

Essentially, emergent abilities enable LLM to perform some degree of complex reasoning. For example, extracting structured knowledge from unstructured text. This ability allows LLM to understand some NLP tasks that are naturally underlying in the text corpus that it was trained on. For simpler tasks, we can think of the LLM with emergent abilities that have already been finetuned during training, and for more complex tasks, it can break them down into simpler tasks. The emergent abilities do not necessarily bless decoder-only models more than ED ones, but they reduce the performance gap achieved by ED models over decoder-only ones with multitask finetuning.

In-Context Learning from Prompts

Another thing to take into consideration is prompting. When using LLM, we could apply prompt engineering methods like providing few-shot examples to help LLM understand the context or task. In this paper by Dai et al. Researchers mathematically proved that such in-context information can be seen to have a similar effect as gradient descent that updates the attention weight of the zero-shot prompt.

$$
\begin{aligned}
\widetilde{\mathcal{F}}_{\mathrm{ICL}}(\mathbf{q}) &= W_{\mathrm{ZSL}}\mathbf{q} + W_V X' \left(W_K X'\right)^T \mathbf{q} \\
&= W_{\mathrm{ZSL}}\mathbf{q} + \mathrm{LinearAttn}\left(W_V X', W_K X', \mathbf{q}\right) \\
&= W_{\mathrm{ZSL}}\mathbf{q} + \sum_i W_V \mathbf{x}'_i \left(\left(W_K \mathbf{x}'_i\right)^T \mathbf{q}\right) \\
&= W_{\mathrm{ZSL}}\mathbf{q} + \sum_i \left(\left(W_V \mathbf{x}'_i\right) \otimes \left(W_K \mathbf{x}'_i\right)\right) \mathbf{q} \\
&= W_{\mathrm{ZSL}}\mathbf{q} + \Delta W_{\mathrm{ICL}}\mathbf{q} \\
&= \left(W_{\mathrm{ZSL}} + \Delta W_{\mathrm{ICL}}\right)\mathbf{q}.
\end{aligned}
\tag{12}
$$

In-context learning adds to the attention weight of a zero-shot learning input (normal prompt)

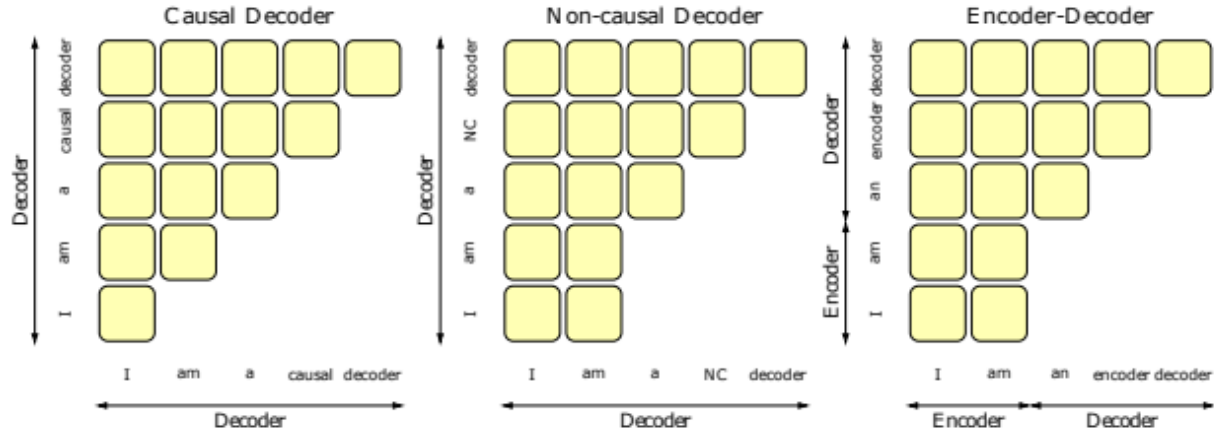
If we perceive prompting as introducing a gradient to the attention weight, we can probably expect it to have a more straightforward effect for the decoder-only models as it does not need to be translated into an intermediate context first before being used for generative tasks. Logically, it should still work for encoder-decoder architectures, but it requires the encoder to be carefully tuned to an optimal performance which might be difficult.

Efficiency Optimization

in decoder-only models, the Key (K) and Value (V) matrices from previous tokens can be reused for subsequent tokens during the decoding process. Since each position only attends to previous tokens (due to the causal attention mechanism), the K and V matrices for these tokens remain unchanged. This caching mechanism improves efficiency by avoiding the recomputation of K and V matrices for tokens that have already been processed, facilitating faster generation and lower computational costs during inference in autoregressive models like GPT.

Autoregressive vs Bidirectional Attention

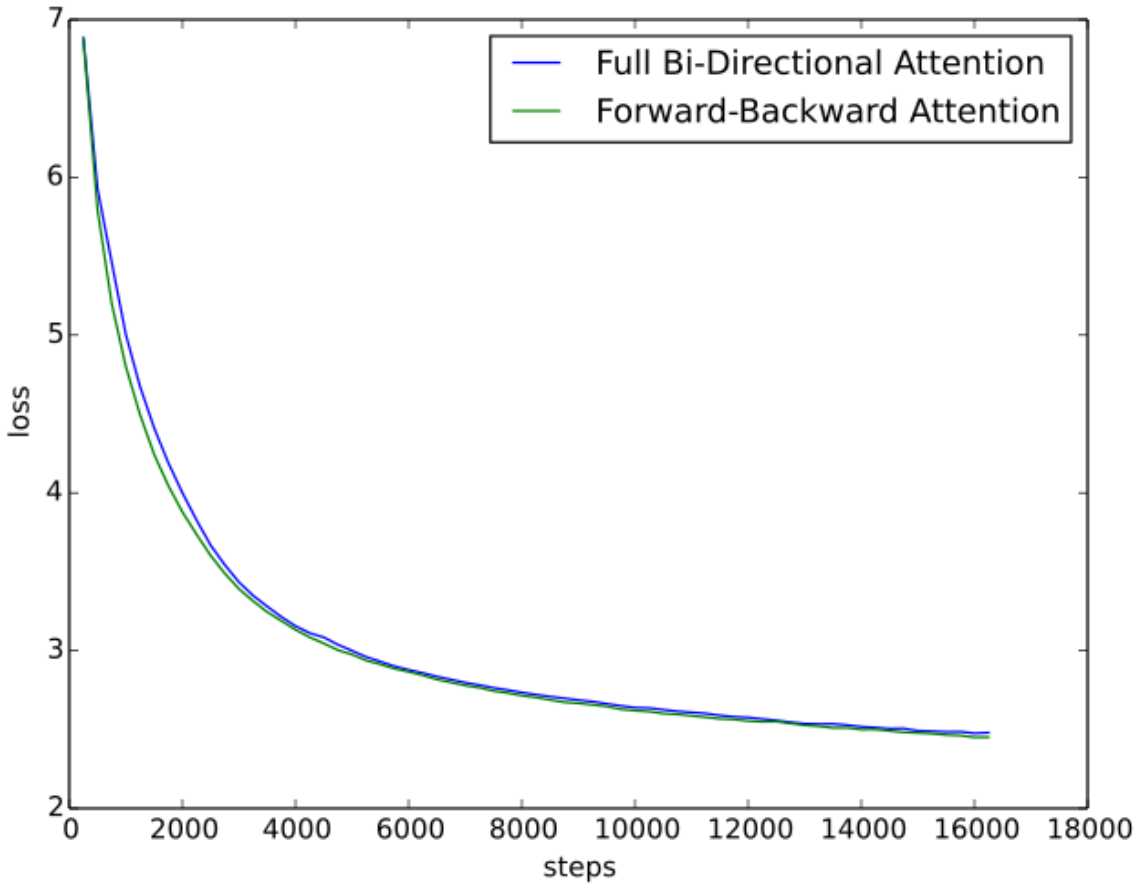
There is another interesting point that is raised regarding the difference in the underlying attention mechanisms, that is, autoregressive for Decoder-only (Causal Decoder) and Bidirectional for Encoder-Decoder. We can visualize how they attend different positions below:



Positions attend differently in these architectures.

The way the attention matrix is calculated in Transformer Architecture is by multiplying two lower-dimensional matrices (Q and K^T) and then applying softmax operation. In decoder-only architectures, the attention matrix is constrained to a lower triangular form due to causal masking (to prevent the model from seeing future tokens), which theoretically maintains its full rank status: Each element in the diagonal (representing self-attention) contributes to making the determinant positive (You only get a positive result from softmax). The full rank status suggests a theoretically stronger expressive capability.

The other two generative architectures both introduce bidirectional attention and thereby do not guarantee the full rank status of their attention matrix. The author suggests that this will limit the performance of the model. He set up an experiment to verify this assumption by splitting the bidirectional attention matrix into one-directional, with half the attention heads attending forward and the other half backward. Then he compared the performance of such Forward-Backward attention to the Full-bidirectional attention model. The FB attention did perform better which kind of verified this theory, but the improvement was quite marginal and did not seem to suggest a significant difference especially when the models are sufficiently trained.



Performance of FB Attention compared to Full Bi-Directional

It intuitively makes sense. Bidirectional attention works as a double-edged sword: It speeds up the learning process but also kind of “spoils” the model from learning the deeper predictive patterns essential for generation. You can kind of think of it as learning how to write: filling in blanks is easier compared to writing an entire article word by word, but it would be a less effective way of practicing. However, after enormous amount of training, both approaches achieve the objective of learning how to write.

What did I learn

The popularity of decoder-only architecture comes from its simplicity, good zero-shot generalization, and cheaper training cost to attain a reasonable performance. Many works have been done studying the performance of decoder-only and encoder-decoder architectures, but given there is sufficient training and model size, there really is no hard evidence that proves one architecture is superior to another in terms of final performance.

In fact, Google Gemini showed just how the encoder-decoder model can work just as well and even exceed decoder-only architectures in some tasks. The encoder component supports the “build-in multimodality” by enabling extracting information from non-textual inputs which might be crucial for the future generation of LLMs. Our initial question truly should be — why WERE most LLMs decoder-only — it showed an era where everyone had mainly worked on advancing decoder-only architectures. Nonetheless, I think it still reveals a great amount of insights into understanding the inner mechanism of how LLMs work and the history of their advancement. It’s exciting to see what will come next in the search for AGI.

Reference

Dai, D., Sun, Y., Dong, L., Hao, Y., Sui, Z., & Wei, F. (2022). Why Can GPT Learn In-Context? Language Models Secretly Perform Gradient Descent as Meta-Optimizers. Retrieved from https://arxiv.org/abs/2212.10559

The BigScience Architecture & Scaling Group. (2022). What Language Model Architecture and Pretraining Objective Work Best for Zero-Shot Generalization? Retrieved from https://arxiv.org/abs/2204.05832

Fu, Y., Khot, T., & Peng, H. (2022, November 20). A Closer Look at Large Language Models Emergent Abilities. Retrieved from https://yaofu.notion.site/A-Closer-Look-at-Large-Language-Models-Emergent-Abilities-493876b55df5479d80686f68a1abd72f

Wei et al. (2022). Emergent Abilities of Large Language Models. Retrieved from https://arxiv.org/abs/2206.07682

Dong, Y., Cordonnier, J.-B., & Loukas, A. (2021). Attention is not all you need: pure attention loses rank doubly exponentially with depth. Retrieved from https://arxiv.org/abs/2103.03404

苏剑林. (Mar. 17, 2023). 《为什么现在的LLM都是Decoder-only的架构？》[Blog post]. Retrieved from https://kexue.fm/archives/9529

Alammar, J (2018). The Illustrated Transformer [Blog post]. Retrieved from https://jalammar.github.io/illustrated-transformer/