MY WEIRD PROMPTS

Podcast Transcript

EPISODE #267

Decoding the Transformer: From Attention to Inference

Published January 21, 2026 • Runtime: 19:37

https://myweirdprompts.com/episode/transformer-inference-architecture-evolution/

EPISODE SYNOPSIS

In this episode, Herman and Corn break down the "black box" of the transformer architecture, moving beyond the 2017 "Attention Is All You Need" paper to explore how modern LLMs actually process data during inference. They discuss the critical shift from encoder-decoder models to decoder-only giants, the memory-saving brilliance of KV caching, and the hardware-aware speed of FlashAttention-3. From speculative decoding to Rotary Positional Embeddings, learn how these technical plumbing upgrades have transformed simple translation tools into sophisticated world models capable of reasoning. This deep dive covers the journey of a token from a numerical vector to a human-readable response, revealing the complex engineering that powers today's most advanced AI systems.

DANIEL'S PROMPT

Daniel

I’d like to talk more about the mechanism of the transformer, specifically the encoding and decoding aspects. How do the encoding and decoding processes of inference work exactly, and what have the advancements been since the transformer architecture was introduced?

TRANSCRIPT

Corn

Hey everyone, welcome back to My Weird Prompts. I am Corn, and I am sitting here in our living room in Jerusalem with my brother, looking out at a rather gray afternoon.

Herman

And I am Herman Poppleberry. It is a perfect day for staying inside and talking about some serious technical architecture. Our housemate Daniel sent us a prompt this morning that really gets into the gears of how the modern world works. He wants to dig into the transformer mechanism, specifically the encoding and decoding aspects of inference, and how things have changed since that landmark paper came out in two thousand seventeen.

Corn

It is a great question because we use these terms so often—encoder, decoder, transformer—but I think for a lot of people, the actual movement of data during inference is a bit of a black box. We know it takes text in and spits text out, but the handoff between the numerical representation of a concept and the actual prediction of a word is where the magic happens.

Herman

Exactly. And Daniel pointed out something important in his prompt—this idea that we are moving between a human world of text and a matrix world of numbers. It really is like the movie The Matrix, where you are looking at the code behind the curtain. But instead of falling green characters, it is massive high dimensional vectors.

Corn

So, let us start with the basics of the original architecture. When people talk about the transformer, they are usually referencing the Attention Is All You Need paper. Back then, the standard was a balanced encoder-decoder structure. Herman, for the sake of setting the stage, how did that original split work during a task like translation?

Herman

Right, so in that original setup, the encoder was the part of the model that looked at the entire input sequence at once. If you were translating a sentence from English to Hebrew, the encoder would take the whole English sentence, process it through multiple layers of self-attention, and create a rich numerical representation where every word is understood in the context of every other word in that sentence. It is essentially building a map of meaning.

Corn

And that is a parallel process, right? The encoder is not reading word by word like a human; it is looking at the whole block.

Herman

Exactly. It is highly efficient on hardware because you can compute all those relationships simultaneously. But then you have the decoder. The decoder is the creative side. Its job is to generate the output, like the Hebrew translation, one token at a time. This is what we call auto-regressive generation. It takes the context from the encoder and says, okay, based on this English meaning, what is the first Hebrew word? Then it takes that first word, feeds it back into itself, and asks, what is the second word?

Corn

This is where I want to push a bit deeper on the inference side. Most of the models we use today, like GPT-five or the Llama four series, are actually decoder-only. We have moved away from that balanced split for a lot of general purpose tasks. Why did that happen, and how does the encoding step work when there is no dedicated encoder?

Herman

That is a brilliant catch, Corn. In a decoder-only model, the encoding and decoding are essentially happening in the same block of layers. When you type a prompt into a chatbot, the model first performs a pre-fill phase. This is essentially the encoding step. It takes your entire prompt, calculates the hidden states for every token in that prompt, and stores them. It is still doing that contextual mapping, but it is doing it using the same causal mask that it will use for generation.

Corn

So, during that pre-fill, it is still parallel? It is looking at my whole paragraph at once to understand the intent?

Herman

Yes, it is very fast. Your graphics processing unit can chew through that prompt in one or two clock cycles depending on the length. But once it has finished encoding your prompt into its internal memory, it switches to the decode phase. And this is where the bottleneck is. It has to generate the next token, then wait, then generate the next. It is sequential. You cannot parallelize the future.

Corn

That makes sense. You cannot predict the third word until you have decided what the second word is. But this brings up a massive technical challenge that I know you have been reading about lately—the Key-Value cache, or KV cache. Most people do not realize that the reason these models can respond so quickly is because of a very clever memory trick. Can you break that down? Because to me, that is the heart of modern transformer inference.

Herman

Oh, I would love to. The KV cache is arguably the most important piece of engineering in modern large language model deployment. Think about it this way: if the model has to look at the entire conversation every time it generates a new word, the amount of math it has to do would grow exponentially. By the time you are at the end of a long essay, it would be recalculating the meaning of the first sentence for the ten thousandth time.

Corn

Which would be a massive waste of electricity and time.

Herman

Precisely. So, the KV cache is a way of saying, hey, we already calculated the key and value vectors for these previous words in the encoder phase. Let us just save those in the memory of the graphics card. When we generate the next word, we only calculate the math for that one new word and then look back at our saved notes for everything else. This turns a problem that would take quadratic time into a problem that takes linear time.

Corn

But there is a catch, right? Memory is not infinite. We are seeing these models now, like Llama four Scout, with context windows of ten million tokens. If you are caching all those vectors for every single layer of a model that has hundreds of billions of parameters, you are going to run out of video random access memory very quickly.

Herman

You are hitting on the exact reason why model efficiency is the biggest topic in AI research right now. A model with four hundred billion parameters, if you are storing the KV cache for a massive context window, you might actually need more memory for the cache than for the model itself! This is why we have seen advancements like Grouped Query Attention. Instead of every single attention head having its own set of keys and values, they share them. It reduces the memory footprint of that cache by a factor of eight or more.

Corn

That is fascinating. It is almost like the model is becoming more efficient by learning to summarize its own internal state. But I want to go back to something Daniel mentioned—the movement between the world of numbers and the world of text. When the decoder finally decides on a token, it is not actually picking a word, right? It is outputting a probability distribution over the entire vocabulary.

Herman

Right. It is a vector that is as long as the vocabulary—maybe thirty-two thousand or even a hundred and twenty-eight thousand entries. Each entry is a number representing how likely that word is to come next. Then we use a process like top-p sampling or temperature to actually pick one. This is the final step of the decoding process. And it is actually quite a small part of the computation, but it is the part where the personality of the AI comes out. If you set the temperature high, it might pick a less likely word, making it more creative or weird.

Corn

I love that. It is the ghost in the machine moment. But let us talk about the advancements since two thousand seventeen. The original transformer used something called sinusoidal positional encodings. But that has changed, hasn't it? We have moved on to things like Rotary Positional Embeddings, or RoPE, and now even interleaved RoPE in the newest models. Why did the original way fall short?

Herman

The original sinusoidal method had limitations with extrapolation. It was like a ruler that only worked well within a certain range. If you trained the model on sequences of five hundred tokens, it would struggle with sequences significantly longer than that, with performance degrading as you moved beyond the training distribution. Rotary Positional Embeddings are much more elegant. Instead of just adding a number to the word, they rotate the vector in a high dimensional space. The angle of the rotation tells the model the relative distance between words. This is huge for inference because it allows the model to generalize to those ten million token sequences we see today.

Corn

That is a great analogy. It is like knowing that the kitchen is three steps from the living room, regardless of where you are in the house. But we should also talk about the speed of inference. Daniel asked about how it works exactly, and a huge part of the modern process is something called FlashAttention-3. Have you come across that?

Herman

I have seen the name, and I know it is supposed to be a massive speedup, but I will be honest, the actual mechanism of how it interacts with the hardware is a bit fuzzy for me. It is about how the data moves between different types of memory on the chip, right?

Corn

Yes, it is a triumph of hardware-aware engineering. FlashAttention-3, released in 2024, reordered the math so the chip can calculate the attention in small blocks that fit entirely in the fast, local memory of the graphics chip. It never has to write the big intermediate matrix to the slow main memory. It is like a chef bringing a small spice rack right to the stove instead of running to the basement for every pinch of salt. It makes the process significantly faster than the previous version and uses much less memory.

Herman

So, we have more efficient memory with the KV cache, better positional understanding, and faster math. But even with all that, these models are still slow compared to a traditional search engine. This brings us to speculative decoding. This one blew my mind when I first heard about it. It is almost like the model is guessing its own future, right?

Corn

It is exactly that. Imagine you have a giant, slow, brilliant model like GPT-five. And you have a tiny, fast, slightly dumber model. In speculative decoding, the tiny model guesses the next five or six words. Then, the big model looks at those six words all at once—remember, the pre-fill phase is parallel! It checks the tiny model's work. If the tiny model got four out of six words right, the big model keeps those four and only has to do the heavy lifting for the ones it wants to change. You get the intelligence of the big model but the speed of the small model for the easy parts of the sentence.

Herman

We have talked a lot about the technical side, but I want to take a step back and think about what this means for the user. When Daniel asks about the encoding and decoding process, he is really asking how these machines think. And what strikes me is that the transition from the transformer being a translation tool to being a general reasoning engine happened because we realized that the decoder is actually a world model.

Corn

That is a deep point, Herman. In the beginning, we thought we needed the encoder to understand and the decoder to speak. But as it turns out, to be a really good predictor of the next word, you have to understand everything. The decoding process is not just picking words; it is simulating a path through the space of all possible ideas. And that brings us to Mixture of Experts, or MoE. This is how large models can have hundreds of billions of parameters but still be fast. They only activate a small fraction of those experts for any given word.

Corn

It feels like we are in this era where the basic architecture of the transformer is staying the same, but the surrounding plumbing is being completely reinvented. We went from the simple encoder-decoder of two thousand seventeen to this incredibly complex mesh of speculative decoding and mixture of experts.

Herman

And do not forget the shift in how we handle the input itself—the tokenization. Early on, we had very simple ways of breaking words into pieces. Now, we have much more sophisticated byte-pair encoding that can handle multiple languages and even code much more efficiently. If your tokenizer is better, your encoder has to do less work because the numerical representation starts off closer to the actual meaning.

Corn

I think it is important for people to realize that when they are using an AI in twenty-six, they are not just using a clever piece of software. They are using a stack of mathematical innovations that have solved some of the hardest problems in computer science in record time. The move from quadratic to linear complexity in attention, the mastery of the memory bottleneck—these are the things that made the AI revolution actually usable.

Herman

It is true. If we were still using the exact same inference methods from two thousand seventeen, these models would be so slow and expensive that only governments could afford to run them. The fact that you can run a decent model on a high-end laptop today is a testament to the engineering that happened after the initial breakthrough.

Corn

So, looking forward, do you think the transformer remains the dominant architecture? Or are we going to see a shift toward something else, like State Space Models?

Herman

That is the million dollar question. We are seeing models like Jamba—which was actually developed right here in Israel by AI21 Labs—that use a hybrid approach. They put transformer layers on top of a state space model. It gives you the reasoning power of a transformer but the efficiency of a model that doesn't have the same KV cache bottleneck. We are essentially watching the evolution of digital brains in real time.

Corn

Well, I think we have given Daniel a lot to chew on. From the basic split of the encoder and decoder to the wizardry of FlashAttention and speculative guessing. It is a lot more than just predicting the next word. It is a massive orchestral performance of data moving through silicon.

Herman

It really is. And I hope this helps our listeners visualize what is happening the next time they see that little cursor blinking while an AI generates a response. There is a lot of hard-earned math in those pauses.

Corn

Definitely. And hey, if you are out there listening and you find this kind of deep dive interesting, we would really appreciate it if you could leave us a review on your podcast app or on Spotify. It genuinely helps other curious people find the show.

Herman

It really does. We love doing these deep dives, and knowing people are out there geeked out on KV caches with us makes it all the better.

Corn

You can find all our past episodes, including our deep dive into the history of AI, at myweirdprompts.com. We have an RSS feed there and a contact form if you want to send us a prompt like Daniel did.

Herman

Or if you just want to tell us your favorite type of positional encoding. We are open to all levels of nerdery here.

Corn

Speak for yourself, Herman Poppleberry. Anyway, thanks for joining us today in Jerusalem. This has been My Weird Prompts.

Herman

Until next time, keep asking the weird questions.

Corn

Take care, everyone.

Herman

Goodbye!

Corn

So, Herman, be honest. If you were a transformer, would you be an encoder or a decoder?

Herman

Oh, I am definitely a decoder. I can't stop talking once I start. I am very auto-regressive.

Corn

I knew you were going to say that. I think I would be more of the attention mechanism—trying to figure out which parts of what you said are actually important.

Herman

And you do a great job of it, brother. Alright, let us go see if Daniel wants some tea.

Corn

Sounds like a plan. See you all in the next one.