


★ Member-only story

LLMs For Time Series Forecasting !!!



Vishal Rajput  · [Follow](#)

Published in AlGuys

10 min read · Nov 26, 2024



Listen



Share

... More

If you take a look at the industrial data you would see that in many places we are still using classical Machine Learning algorithms. There is a good reason to use classical ML and AI algorithms over new Deep learning-based methods in industrial settings; the amount and quality of proprietary data. Most banks still use some variant of XGBoost for tabular data. We have seen crazy progress in Deep Learning models, but there are still many fields where growth has been barely linear. One such field where we have seen limited growth is time series forecasting.

Topics Covered

- Understanding Time Series Data
- Turing Completeness
- Other Challenges And Approaches
- Foundational Models As Zero-Shot Forecasters (**Chronos** and **TimesFM**)
- Successor of LSTM: xLSTM
- Conclusion

Understanding Time Series Data

Time series data is one of the most naturally occurring forms of data. From **internet usage to smart bands**, from **weather data to the stock market**, everything comes under the umbrella of the Time series.



Time Series Data Sample

Time series is similar to predicting the future based on past events for a given number of quantities/attributes.

Time series data exhibit dependencies across time steps, known as autocorrelation. Capturing these dependencies requires models to understand both short-term and long-term relationships within the data.

Capturing long-term dependency is something that most Deep-learning models are not particularly good at. Now some of you might say what about LSTMs and RNNs, and you are right. They are indeed good for time series type of data.

But we should understand that these models are very different than CNNs and Transformers, they are actually **Turing Complete** and that's why they can handle time series.

Turing Completeness

If you are a Computer science grad, you should definitely understand what Turing Complete means. This forms the theoretical basis of what type of system can solve what type of challenges.

Turing machines are theoretical computers defined by Alan Turing in his highly influential paper titled *On Computable Numbers, with an application to the Entscheidungsproblem*, published 1936. Turing machines are abstract mathematical constructs that help us describe in a rigorous fashion what we mean by *computation*.

A Turing machine consists of 2 elements: The computational *head* and an infinitely long *tape*. The head operates roughly as a ‘read-write’ head on a disk drive, and the tape is divided up into an infinitely long set of squares, for which on each square a symbol can be written or erased. The Turing machine recognizes and can write down a finite set of symbols, called the Turing machine’s *alphabet*.

The Turing machine is only ‘aware’ of one square on the tape at a time — namely, the square the head of the Turing machine is currently on.

On that tape, a Turing Machine can do any of these 4 actions:

- Move the head left by 1 space
- Move the head right by 1 space
- Write a symbol at the head
- Erase a symbol at the head

The machine decides which of these operations to do on any given step through a *finite state machine*. Different Turing machines have different state machines that define their operation.

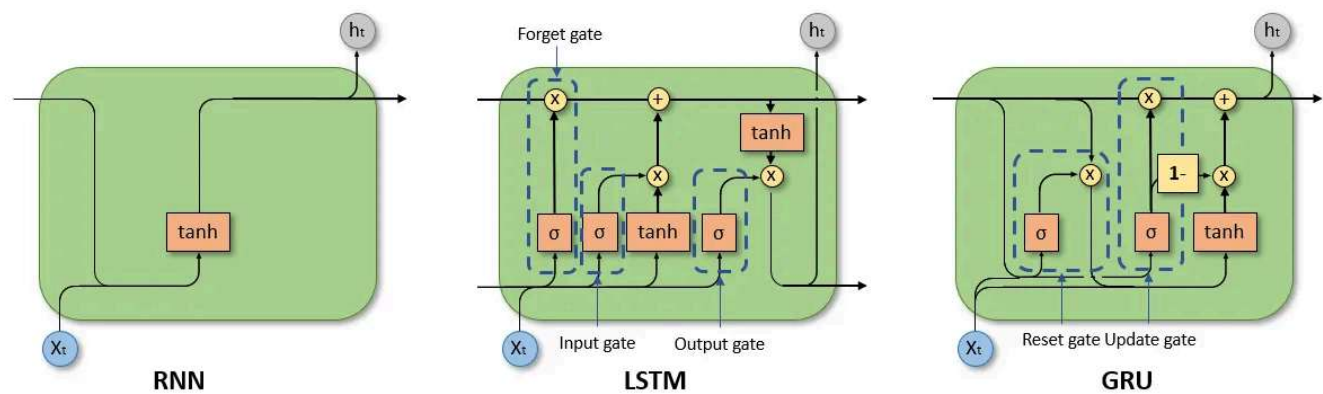
This is all fine theory-wise, but how does it relate to our problem of time series?

A model is Turing complete if it can simulate any Turing machine, implying it can compute anything that is computable, given sufficient time and resources.

Since models don’t have infinite memory, they need to remember and forget a few things, and this level of control is not that easy with Deep learning models. Most

Deep-learning models will remember short-term interactions and are not very good at handling long-term dependencies.

And that’s where LSTM and RNNs shine. RNNs have their own limitations, but the LSTM was king in time series for quite some time. We will also see what happened to LSTMs in a bit.



Different types of deep learning architecture exploit different types of symmetries and except LSTM most are not designed for Time series type of data.

Architecture	Domain Ω	Symmetry group \mathfrak{G}
CNN	Grid	Translation
Spherical CNN	Sphere / $SO(3)$	Rotation $SO(3)$
Intrinsic / Mesh CNN	Manifold	Isometry $Iso(\Omega)$ / Gauge symmetry $SO(2)$
GNN	Graph	Permutation Σ_n
Deep Sets	Set	Permutation Σ_n
Transformer	Complete Graph	Permutation Σ_n
LSTM	1D Grid	Time warping

Transformers which these days form the backbone of all the major AI models, are not inherently designed for time-related stuff.

If you want to know more about this:

Wikipedia: [Turing Completeness](#)

What exactly is Turing Completeness?

An armory for the insatiable pedant

evinsellin.medium.com

Other Challenges And Approaches

Deep learning models typically require huge amounts of data to train effectively. In many time series applications, especially those with limited historical data, classic methods perform adequately without the need for extensive datasets.

Time series data often exhibit non-stationary behavior, seasonality, and trends. Preparing such data for deep learning can be more challenging compared to classic methods that are inherently designed to handle these characteristics. Another big challenge in Time Series Data is to align two-time series sequences.

Tricks in Time Series Data Processing

Time series data is one of the most naturally occurring forms of data. From internet usage to smart bands, from weather...

medium.com

Another big challenge is that Deep learning models can easily overfit and are black box in nature. Being able to explain the prediction might be as important as actually predicting in a few cases.

All these reasons combined, we see far fewer Deep learning models in time series compared to image and text modality.

In the past, we tried different models like Temporal Convolutional Networks (TCNs), Transformer-based models, and hybrid approaches combining classic and deep learning methods, but their generalization and accuracy remained questionable.

But now that we have LLMs that have seen enormous amounts of data, practically the entire internet, we can actually have good Transformer-based time series models.

Foundational Models As Zero-Shot Forecasters

LLMs or foundation models as Zero-Shot has various advantages.

1. The training set is broad enough, that it should have reasonable performance on unseen data regardless of their granularity, frequencies, sparsity, and perhaps even distribution.
2. You don't need to wait until you have enough data to train a model from scratch (eg. ARIMA but might also apply to a global model such as XGBoost).
3. When the data starts to come in, you can fine-tune the zero-shot model to your domain (or |other purposes, eg, conformal predictions).

Here are some dataset for training zero-shot forecasters.

Salesforce/lotsa_data - Datasets at Hugging Face We're on a journey to advance and democratize artificial intelligence through open source and open science. huggingface.co	
Monash Forecasting Repository @InProceedings{godaehewa2021monash, author = "Godaehewa, Rakshitha and Bergmeir, Christoph and Webb, Geoffrey I. and... forecastingdata.org	

Unfortunately, unlike NLP we don't have good datasets for Time Series evaluation.

To solve the data issue, Synthetic data comes in. Generating synthetic data for time series is relatively easier compared to images and videos.

Chronos

"Chronos: Learning the Language of Time Series" is not a single model but it introduces a framework that leverages transformer-based language models for probabilistic time series forecasting.

The core innovation of Chronos is tokenizing continuous time series data into discrete tokens, enabling the application of language modeling techniques.

Tokenization of Time Series:

- **Scaling:** Each time series is normalized by dividing its values by the mean of their absolute values, ensuring consistency across different scales.
- **Quantization:** The scaled values are discretized into a fixed number of bins, converting continuous data into a sequence of tokens.

Chronos utilizes the T5 transformer architecture, with model sizes ranging from 20 million to 710 million parameters. The vocabulary size is adjusted to match the number of quantization bins.

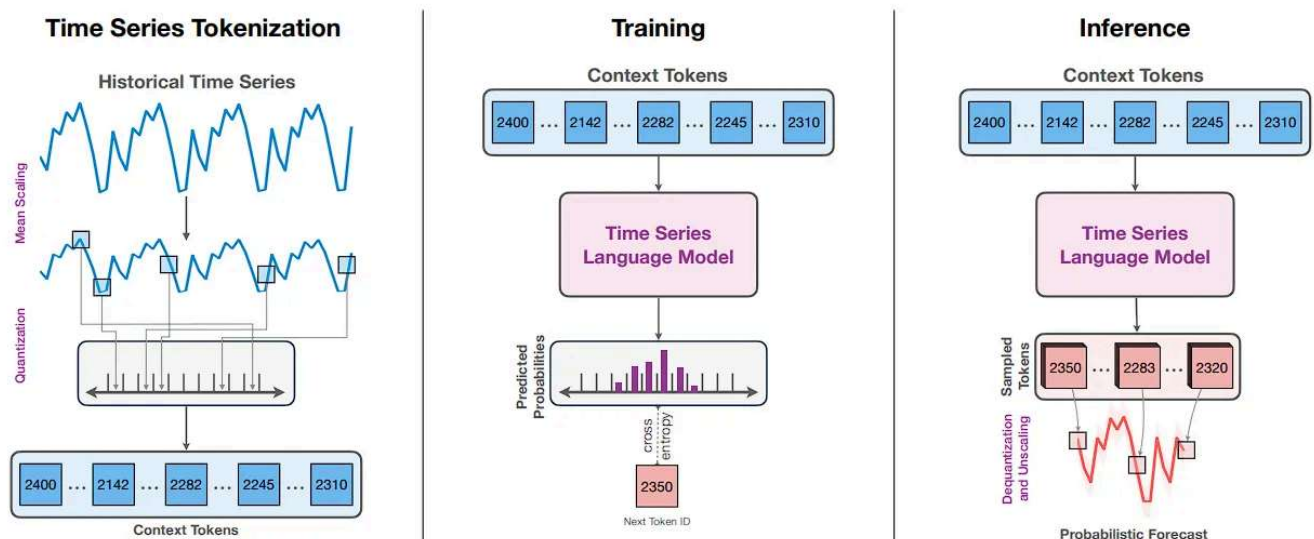


Figure 1: High-level depiction of CHRONOS. **(Left)** The input time series is scaled and quantized to obtain a sequence of tokens. **(Center)** The tokens are fed into a language model which may either be an encoder-decoder or a decoder-only model. The model is trained using the cross-entropy loss. **(Right)** During inference, we autoregressively sample tokens from the model and map them back to numerical values. Multiple trajectories are sampled to obtain a predictive distribution.

The model is trained using the cross-entropy loss function, treating the forecasting task as a sequence prediction problem. Training data comprises a large collection of publicly available time series datasets, supplemented by synthetic data generated via Gaussian processes to enhance generalization.

By framing time series forecasting as a language modeling problem, Chronos simplifies the forecasting pipeline. Its ability to generalize across diverse datasets without additional training positions it as a versatile tool for various forecasting

applications. But as I said time series is a bit different than language prediction. But this is an interesting approach nonetheless.

TimesFM

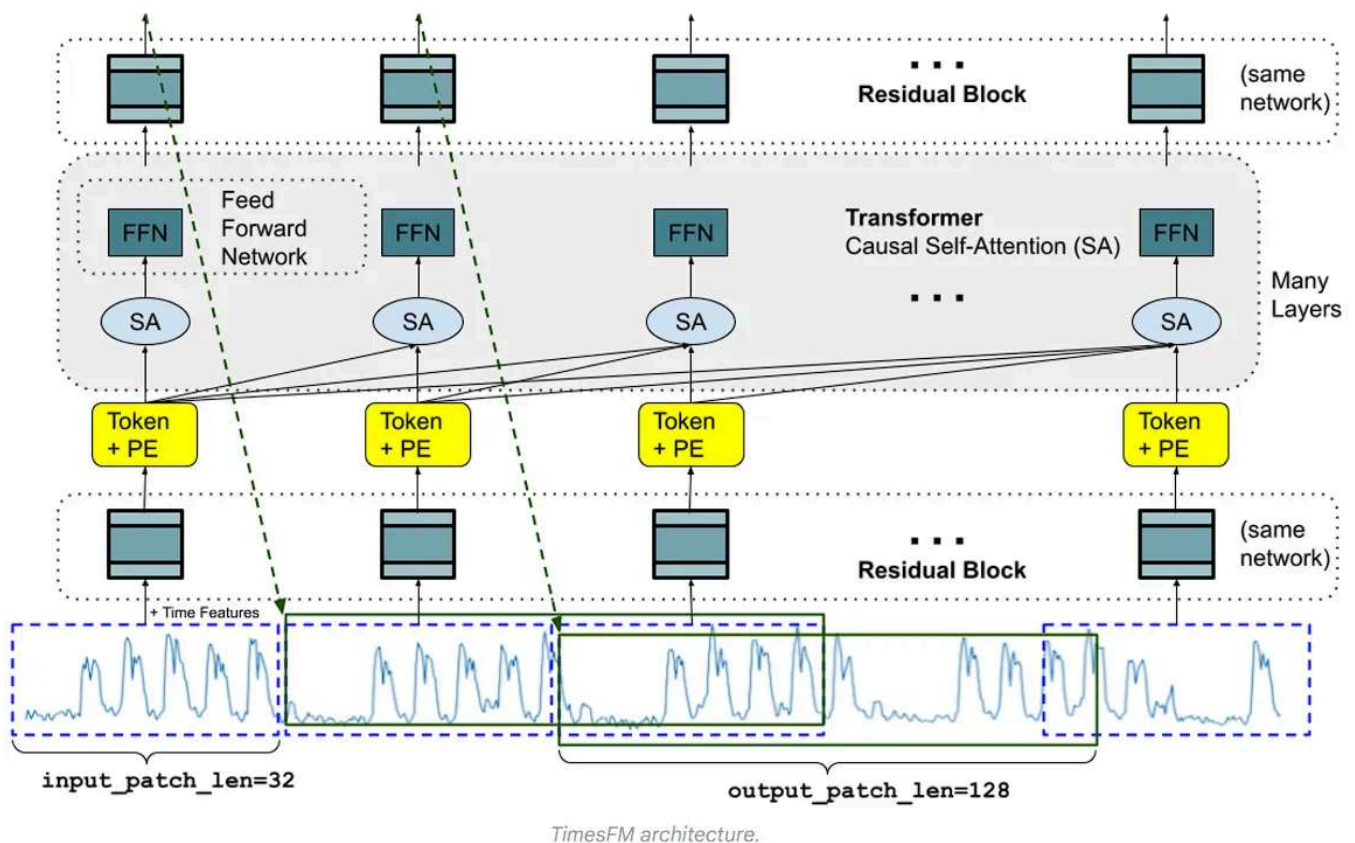
This is a closed-source model from Google, but this is a completely new architecture designed specifically for time series.

TimesFM is a forecasting model, pre-trained on a large time-series corpus of 100 billion real-world time-points, that displays impressive zero-shot performance on a variety of public benchmarks from different domains and granularities.

<p>A decoder-only foundation model for time-series forecasting</p> <p>Posted by Rajat Sen and Yichen Zhou, Google Research Time-series forecasting is ubiquitous in various domains, such as...</p> <p>research.google</p>	
---	--

Chronos converts continuous time-series values into discrete **tokens** through normalization and quantization. These tokens are then fed to the transformer layers.

On the other hand, **TimesFM** uses a different approach. TimesFM treats a **patch** of time series (a contiguous group of time points) as a token. The patches are processed using an MLP with residual connections before being input to the transformer. It focuses on learning temporal patterns directly from the continuous time-series data without discretization.



Instead of doing token-by-token prediction, TimesFM is explicitly designed for **long-horizon forecasting** by using larger output patches, which reduce the number of sequential prediction steps needed, improving both efficiency and accuracy.

But let's go a little deeper, you might still have doubts about TimesFM workings.

Why We Can't Directly Feed Time-Series Patches into Transformers?

Transformers Expect Fixed-Dimensional Inputs:

- Transformers process sequences of tokens, where each token is typically a fixed-length vector.
- Raw time-series patches are matrices (e.g., *patch length* × *features*) and vary in size depending on the patch length and feature count. This is incompatible with the fixed-dimensional token input requirement.

Lack of Feature Aggregation:

Without the MLP block, the transformer would treat each individual time point or feature as a separate token. This can lead to:

- Increased computational overhead.

- Loss of meaningful interactions between features within a patch.

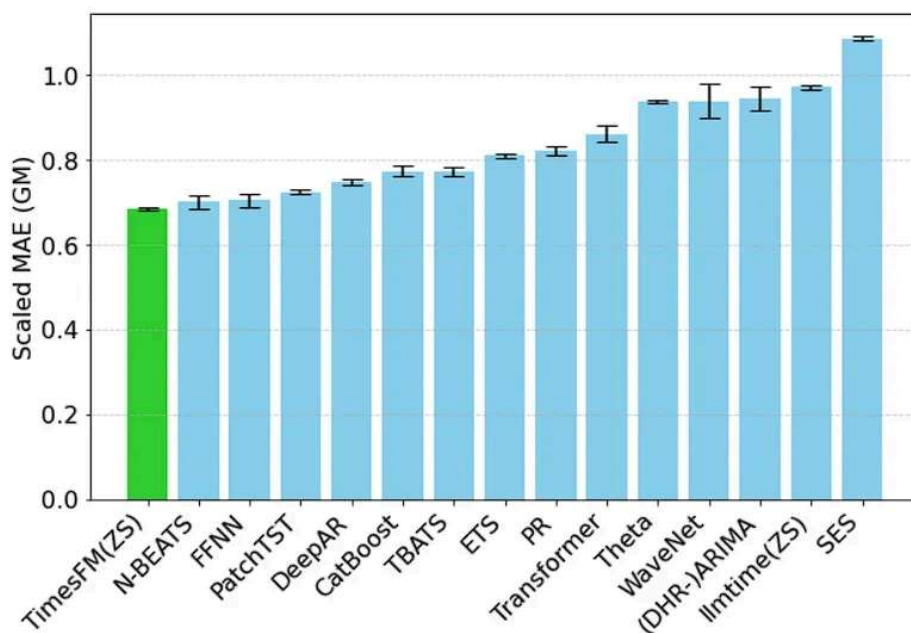
Transformer Computation Costs:

- Transformers scale poorly with sequence length ($O(n^2)$ for attention mechanisms). Feeding raw time-series patches directly would result in excessive computational costs for long time-series data.

Positional Dependency:

- Raw patches do not inherently carry positional information. Without the MLP block incorporating positional encodings, the transformer would not understand the temporal ordering of data points within the patch.

The **MLP block** takes a patch of size $\text{patch length} \times \text{features}$ as input. It processes the patch using linear layers to combine and reduce features, applies nonlinear activations to capture higher-order patterns, and employs residual connections for stability and better gradient flow. The output is a fixed-dimensional vector (e.g., *hidden size*) that serves as a token for the transformer.



[Geometric mean](#) (GM, and [why we do so](#)) of Scaled MAE (the lower the better) of TimesFM(ZS) against other supervised and zero-shot approaches on Monash datasets.

A small table to compare Chronos and TimesFM.

Feature	Chronos	TimesFM
Input Representation	Discrete tokens (via quantization)	Continuous patches (MLP + residuals)
Output Representation	Token-by-token generation	Patch-based generation (variable sizes)
Probabilistic Forecasts	Strong (via sampling)	Less focus on uncertainty
Focus	General-purpose forecasting	Long-horizon efficiency
Architecture	T5-like causal transformer	Decoder-only transformer with patches
Training Data	Public time-series + synthetic	Large-scale real-world + synthetic

Successor of LSTM: xLSTM

LSTMs have three main limitations:

- (i) **Inability to revise storage decisions:** LSTM struggles to revise a stored value when a more similar vector is found, while our new xLSTM remediates this limitation by exponential gating. Basically, if something is forgotten in the previous state somehow, we can't retrieve it, because the memory in LSTM is preserved in some cell state, not the actual information.
- (ii) **Limited storage capacities:** Information must be compressed into scalar cell states. This limitation is exemplified via Rare Token Prediction. LSTM performs worse on rare tokens because of its limited storage capacities. The new xLSTM solves this problem by a matrix memory.
- (iii) **Lack of parallelizability due to memory mixing:** the hidden-hidden connections between hidden states from one time step to the next, which enforce sequential processing. This is the biggest problem of LSTM as to not being able to use the GPU parallelizability to its full extent.

To overcome the LSTM limitations, Extended Long Short-Term Memory (xLSTM) introduces two main modifications. These modifications — **exponential gating and novel memory structures** — enrich the LSTM family by two members: (i) the new **sLSTM** with a scalar memory, a scalar update, and memory mixing, and (ii) the new **mLSTM** with a matrix memory and a covariance (outer product) update rule, which is fully parallelizable.

Both **sLSTM** and **mLSTM** enhance the LSTM through exponential gating. To enable parallelization, the mLSTM abandons memory mixing, i.e., the hidden-hidden recurrent connections. Both **mLSTM** and **sLSTM** can be extended to multiple

memory cells, where sLSTM features memory mixing across cells. Further, the sLSTM can have multiple heads without memory mixing across the heads, but only memory mixing across cells within each head. This introduction of heads for sLSTM together with exponential gating establishes a new way of memory mixing. For mLSTM multiple heads and multiple cells are equivalent.

This new variant of LSTM has shown significant potential once again in time series forecasting, and even beating the foundational model-based Time series models.

Read full article on xLSTM here:

xLSTM vs. Transformers: Who Will Win?

xLSTM revives the age old LSTM with incorporating features from Transformers, making it a serious competitor in the LLM...

medium.com

GitHub - AI-Guru/xlstm-resources: Resources about xLSTM by Sepp Hochreiter

Resources about xLSTM by Sepp Hochreiter. Contribute to AI-Guru/xlstm-resources development by creating an account on...

github.com

Conclusion

Regarding foundational time series models, we are still in the early stage of research. We are yet to explore the effect of fine-tuning, creating better evaluation benchmarks, and much more. Synthetic data is also something that needs to be exploited much more. With this, we mark the end of our blog on Time Series Forecasting.

Thank you for reading! If you enjoyed this article and want more insights on AI, consider subscribing to my newsletter. Get exclusive content, updates, and resources directly to your inbox: <https://medium.com/aiguys/newsletter>

Writing such articles requires considerable effort and time. I would highly appreciate your support through claps and shares. Your engagement motivates me to write more beyond the hype on SOTA AI topics with the utmost clarity and simplicity.

Don't forget to follow me on [X](#).

Time Series Forecasting

AI

Artificial Intelligence

Technology

Machine Learning



Follow

Published in AIGuys

3.4K Followers · Last published 4 days ago



Deflating the AI hype and bringing real research and insights on the latest SOTA AI research papers. We at AIGuys believe in quality over quantity and are always looking to create more nuanced and detail oriented content.



Follow

Written by Vishal Rajput

19.1K Followers · 92 Following

3x  Top writer in AI | AI Book  : <https://rb.gy/xc8m46> | LinkedIn +: <https://www.linkedin.com/in/vishal-rajput-999164122/> | X: <https://x.com/RealAIGuys>

Responses (2)





Adrale

What are your thoughts?



Sumit Bhattacharyya

Nov 29, 2024



Have you looked at Nixtla's TimeGPT? What is your assessment of this?



16



1 reply

[Reply](#)



CARLOS ORTEGA

Dec 26, 2024



Thanks for your good recopilatory document.

Regarding Chronos algorithm, there is a new version smaller, faster and with better forecasting performance ("Chronos-Bolt" - <https://huggingface.co/autogluon/chronos-bolt-base>).

Thanks,

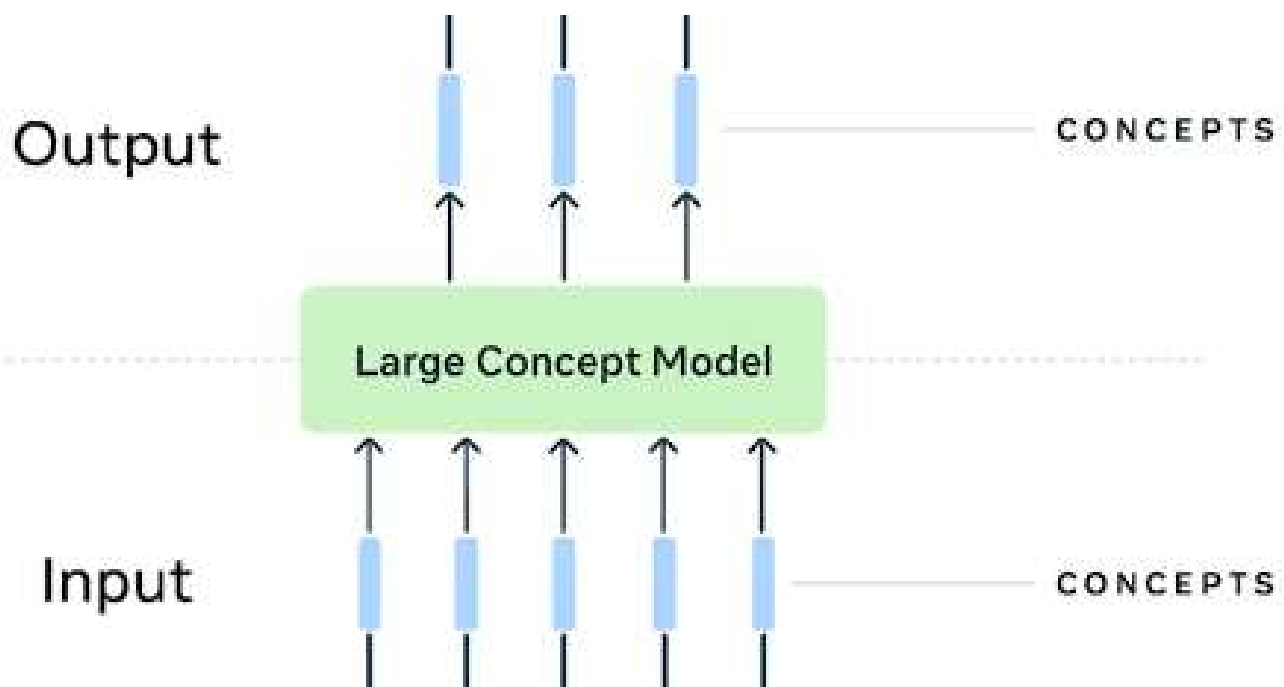
Carlos.



1

[Reply](#)

More from Vishal Rajput and AIGuys



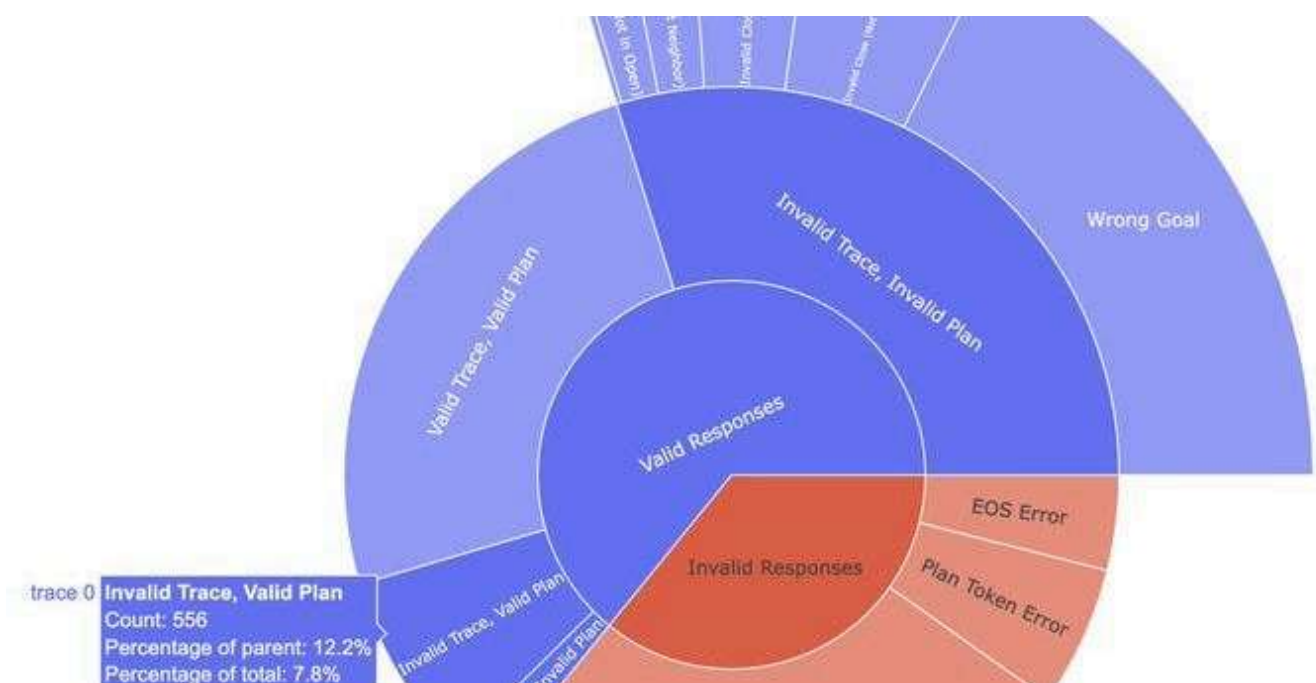
 In AIGuys by Vishal Rajput 

Forget LLMs, It's Time For Large Concept Models (LCMs)

In LCM, modeling is performed in a high-dimensional embedding space instead of on a discrete token representation.

★ Feb 26 🖱️ 284 💬 4

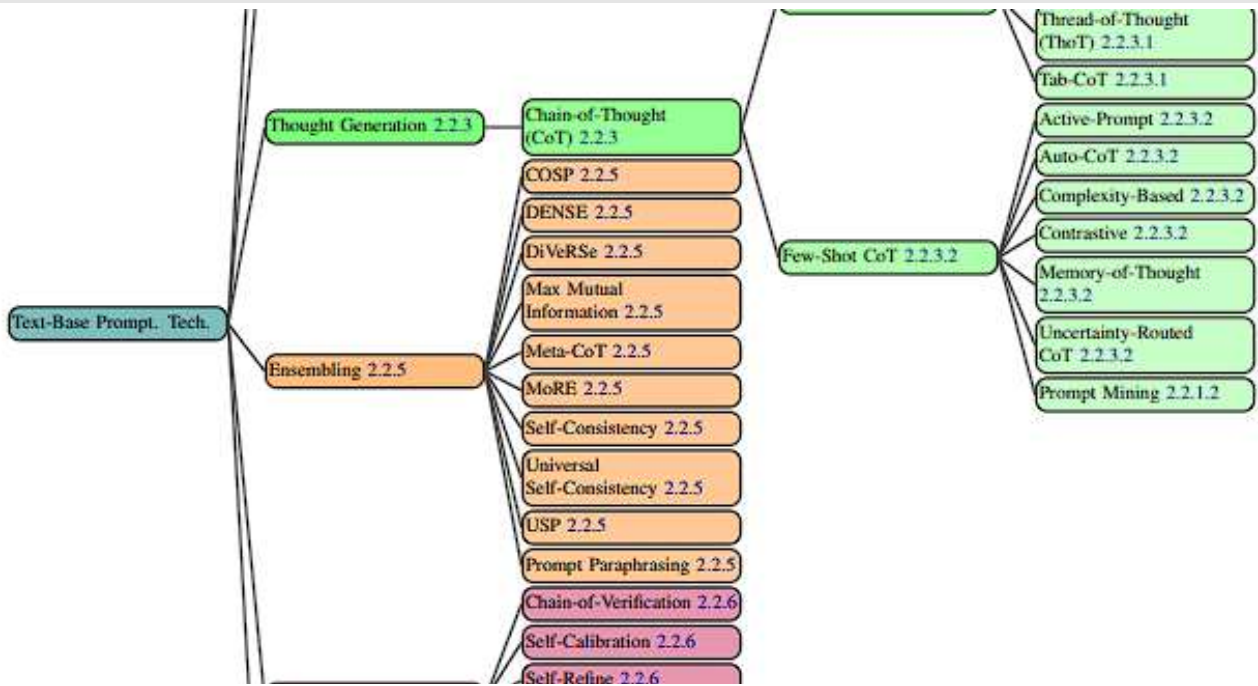
🔖 ⋮



 In AIGuys by Vishal Rajput 

LMMs Not LLMs: Large Mumbling Models

Are we moving towards Large Mumbling Models, but what about Large Reasoning Models?

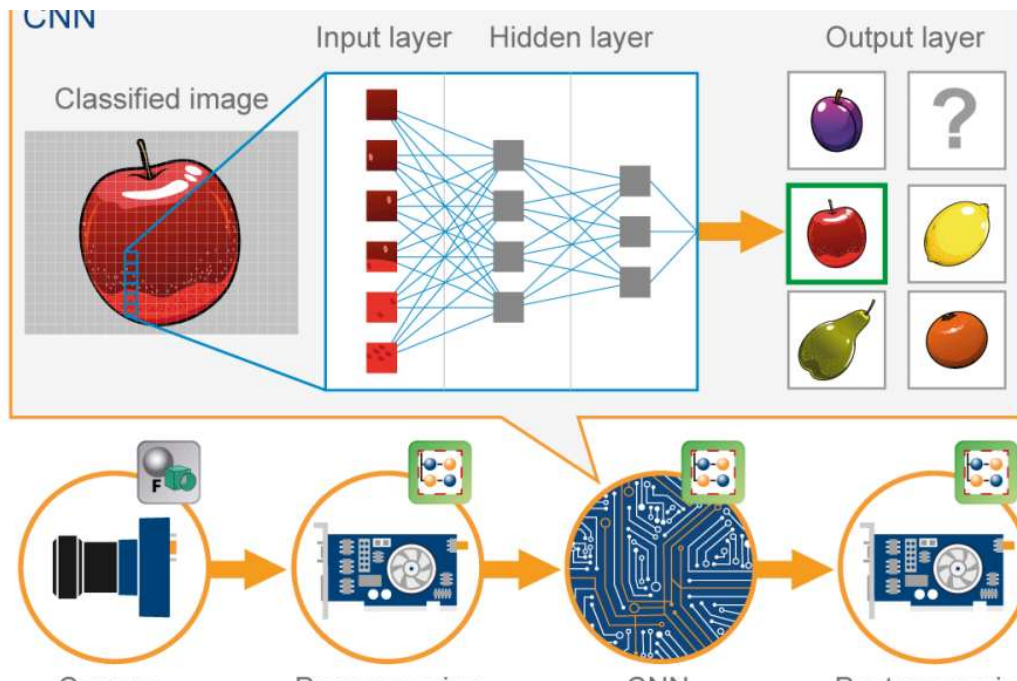


 In AI Guys by Vishal Rajput 🏠

The Prompt Report: Prompt Engineering Techniques

Prompting Techniques Survey

★ Oct 14, 2024 🖱️ 1K 💬 14



 In AI Guys by Vishal Rajput 🏠

Why We Aren't Getting Any Better At AI Alignment?

AI Alignment is not just an engineering problem

See all from Vishal Rajput

See all from AIGuys

Recommended from Medium



TF In The Forecaster by Marco Peixeiro 🏠

Time-LLM: Reprogram an LLM for Time Series Forecasting

Discover the architecture of Time-LLM and apply it in a forecasting project with Python




 In Level Up Coding by Dr. Ashish Bamanian 

Chain-of-Draft (CoD) Is The New King Of Prompting Techniques

A deep dive into the novel Chain-of-Draft (CoD) Prompting that reducing LLM inference cost and latency like never before.

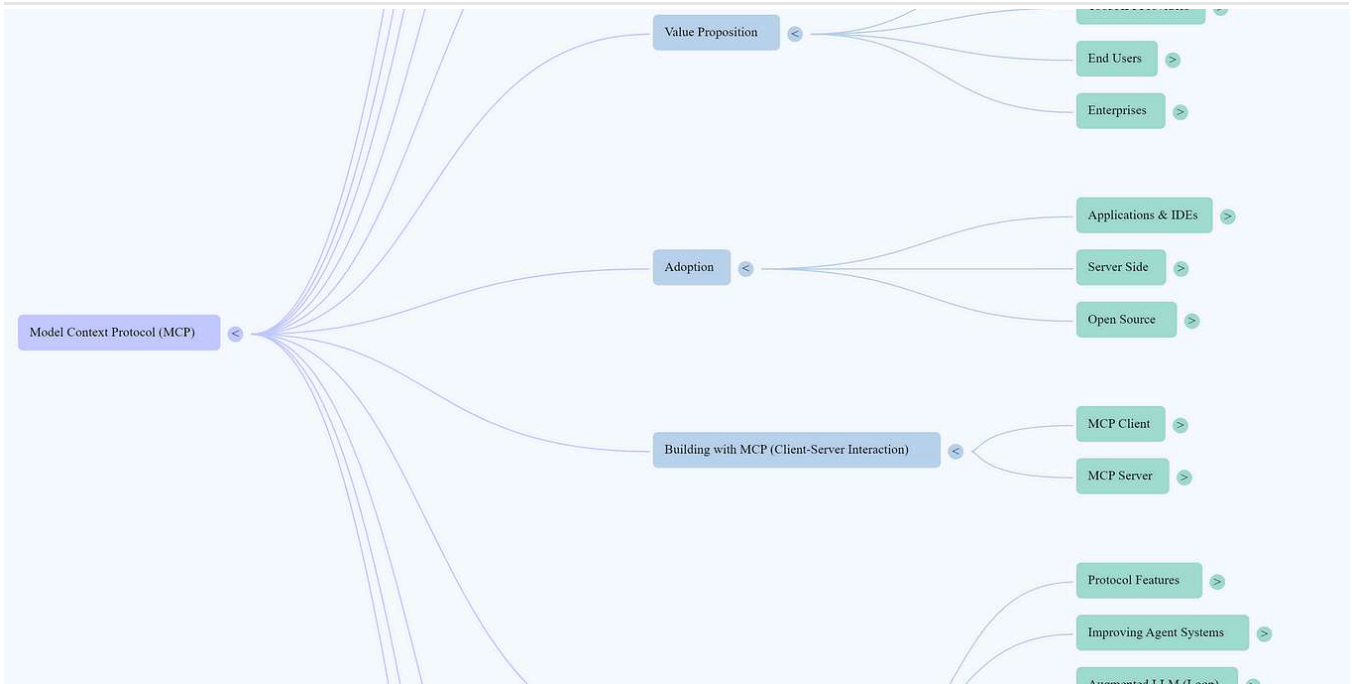
★ Mar 3 🖱 2.3K 💬 35



 Shaw Talebi

Fine-Tuning Text Embeddings For Domain-Specific Search

An overview with Python code

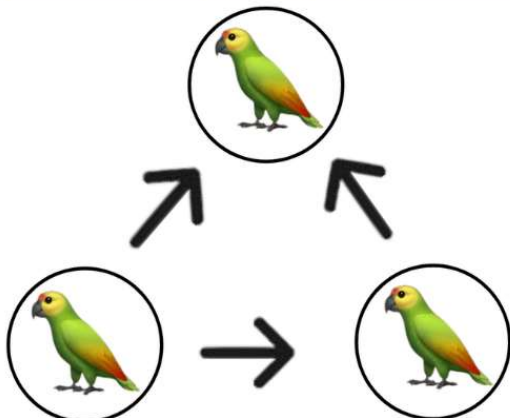


Dr. Nimrita Koul

The Model Context Protocol (MCP)—A Complete Tutorial

Anthropic released the Model Context Protocol(MCP) in Nov. 2024.

Mar 27 421 2



IS INSANE

DSC In Data Science Collective by Gao Dalie (高達烈)

LangGraph + MCP + Ollama: The Key To Powerful Agentic AI

In this story, I have a super quick tutorial showing you how to create a multi-agent chatbot using LangGraph, MCP, and Ollama to build a...

★ Mar 28 🖱️ 673 💬 5

🔖+ ...



📺 In Everyday AI by Manpreet Singh

Goodbye RAG? Gemini 2.0 Flash Have Just Killed It!

Alright!!!

★ Feb 10 🖱️ 3.3K 💬 144

🔖+ ...

See more recommendations