Communications of the ACM

Opinion

Artificial Intelligence and Machine Learning

Forecasting

Avoid large language models for jobs requiring reliable prediction.

By Peter J. Denning

Posted Nov 24 2025

Forecasting is a job we often do with help from computers. A weather forecast uses complex atmospheric models on past weather data to predict the most likely weather conditions in the coming days. A financial forecast uses complex econometric models of markets to specify the most likely financial position of a company in the coming year. Large language models (LLMs) are now being asked to do these jobs. The idea is that LLMs can pore through huge quantities of human text data and locate recurrent patterns that enable them to see farther or deeper than any individual human.

Does this idea hold up? LLMs have been the target of much worry about trustworthiness.[a,b] When can a model's predictions be trusted? A carefully examination of how models in science and engineering are used for predictions shows that many of the claimed powers of AI prediction are illusions reflecting a misunderstanding of prediction. I will walk through the process of validating models for predictions, with the aim of helping you understand whether or when LLMs can be reliable predictors.

What Is a Model?

A model is a representation of a phenomenon. We make models for three main reasons:

1. Explain and understand how the phenomenon works
2. Simulate its behavior
3. Predict how it will behave in the future

In science and engineering, a model is often formulated as a set of equations relating states of components to each other over time. The constraints on component states and connections among them are set by parameters. A good model simplifies by leaving out inessential details without losing fidelity to the phenomenon. It enables explaining, simulating, and predicting faster, cheaper, and safer than direct interaction with the phenomenon.

When we are serious about using a model for prediction, we subject it to validation tests to learn its accuracy. The tests run the model numerous times and calculate the error between the phenomenon's output and the model's output. Validation assumes that we have ways to define and measure the errors. The accompanying figure illustrates the process. The upper tier shows that the phenomenon's responses to provocations can be recorded. The lower tier shows a parameterized model calculating responses to inputs that encode provocations to the phenomenon. Validation is the process of comparing the recorded values to the corresponding calculated values. If they are consistently close, the model is accepted as a substitute to study the phenomenon. A validated model can be used for prediction, but takes some care as we will discuss shortly.

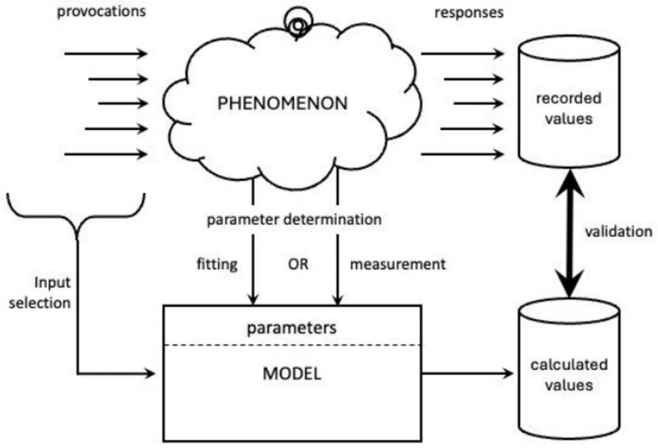

Figure. Model validation.

Parameter determination, at the center of the figure, sets how the model responds. One common method is to measure the parameter values directly in the phenomenon. For example, a queueing network model of a computer network requires, for each server, the mean service time and mean number of times a network job visits it; these values can be measured and supplied to the model. The other common method is to fit the parameters. For example, an LLM has billions of parameters (connection weights) between its neurons; during training, a backpropagation algorithm adjusts parameter values to minimize the error between the model's response to an input and the phenomenon's response.

Sometimes validated models display anomalies by generating unexpected outputs. An accumulation of anomalies calls the model into question and sparks a search for a more accurate, more reliable model.

Example: Validation of Queueing Network Models

Most computer systems and the networks connecting them are meshes of interconnected servers. Uses submit jobs to the network. A job visits a series of servers as it completes its subtasks and finally exits the network. A parameter *N* states the number of jobs in progress at a time. The other parameters say how often a job visits each server and the average service time it receives during each visit. The throughput of the network is its main performance measure.

A queueing network model, or QNM, is a compact mathematical representation of all the physical servers, each with its mean service time and visit count, plus the number of jobs (*N*) and the throughput.[c] Very efficient computational algorithms have been devised to calculate the throughput of the model given its parameters. Extensive validation tests have shown that QNMs reliably compute throughput within 5% of the actual throughput. Most engineers accept this as excellent agreement between the model and real system. They trust a QNM's prediction of the throughput of a network-of-servers system.

Using a model for prediction involves two assumptions. First, the model will still be valid in the future time period. Second, the analyst can reliably project the parameter values for the model in the future period. Parameter projection is a source of uncertainty and error in the model's predictions.

Suppose that a physical network is being reconfigured with new servers, new workload, and a new load distribution algorithm. What will its throughput be? To specify the parameter values for the future configuration, we need to do a careful analysis of the expected future conditions. The mean service times per visit to a server can be taken from the hardware specs of new servers. The visit counts must be adjusted from their current values to reflect changes of the workload characteristics and the load balancing algorithms of the network. Once the parameters are settled, the QNM can then be used to make predictions for future throughput for each workload size *N*.

The uncertainty in projecting parameter values overrides the general conclusion that QNMs predict throughput to within 5%. The only way to know for certain how much additional error was introduced in the parameter projection is to wait until the new system is running and measure its actual parameters and throughput. Computer system engineers go to great pains to be sure they have confidence in the future parameter values, and therefore in the prediction.

This is a problem for any validated model. Parameter projection errors can force the model's calculation to be outside its validation range.

Example: Validating a Large Language Model

Validation is a much more difficult problem with an LLM. For the traditional validation process to work, we would need a precise definition of LLM output for each input. Unfortunately, the main jobs we ask LLMs to do are fuzzy:

1. Engage in conversations
2. Prepare summaries and distillations of many texts
3. Generate new entitles such as essays, poems, or images, given a text description

The difficulties in deciding whether an LLM is doing a good job stem not from the machine, but from the human domain to which we are comparing the machine. It is not obvious how do we measure when a conversation, summary, or creative act is good.

One way we have tried to cope with this to recruit humans to evaluate the goodness of

responses from an LLM. The human valuations are then used to adjust LLM parameters for more satisfactory outputs. This is called reinforcement learning from human feedback (RLHF). To do this, generate a large number of prompts for the LLM. Record the LLM responses in a database of (prompt, response) pairs. Hire humans to evaluate and score the pairs, producing a database of (prompt, response, score) triplets. Train a second model to predict human scores of (prompt, response) pairs. Using the second model and reinforcement learning, retrain the original model to favor responses likely to earn higher scores.

This method has produced some improvements but no guarantees that the LLM outputs will always score well. Moreover, the human scorers add their own biases to the LLM responses.

The Hallucination Problem

The fuzziness of goodness measures is not the only impediment to validating an LLM. Hallucinations are LLM responses that contain nonsense or falsehoods. Hallucination rates for general use of LLMs appear to be under 5% for most LLMs, but they can exceed 70% when LLMs are queried in narrow domains such as law or science. Hallucinations are now accepted as inherent in the design of LLMs. LLMs derive their outputs entirely from the statistics of the training data, with no outside checks of sense or truth.

Hallucinations make it hard to trust LLMs, especially when the output is used to guide decisions where mistakes are costly. A conversation with an AI companion can be comforting up until the companion proposes harmful behavior. A distillation can be useful, but a summary of a critical document may need independent review for accuracy. New images generated for a draft presentations can be problems when the owners of the original images want to be compensated for use of their copyrighted work. A lawyer asking an LLM to generate a case summary must independently check the claimed sources to see if they exist. A military commander using an LLM for recommendations on troop placements can start a war if the LLM gives made-up field information. A voter seeing a compromising video of a candidate cannot be sure the video was faked. A scientist or journalist looking for information to corroborate a claim cannot find it because search engines filter their results through AI summarizers.

Some advocates claim that hallucinations will disappear once larger LLMs incorporate all data on the Internet. This claim does not make sense. The largest LLMs already incorporate the vast majority of those data. Regardless of size, LLMs cannot not tell truth from falsehood.

The quality of data used to train smaller LLMs is another challenge. The data available are being progressively contaminated by massive troves of data deposited on the Internet by LLMs themselves. Many labelled datasets are obtained from low-wage untrained workers who do not know the fields they are labelling in. In other words, the training data themselves are becoming untrustworthy.

LLMs are notoriously bad at responses that demand logical or mathematical reasoning. Even simple problems like adding fractions with different denominators produces

astonishingly bad results. When the user corrects that machine, it typically answers "Good catch!" and fails again to do the arithmetic properly. This may eventually be solved by combining logic machines with LLMs, an active area of research.

Obviously, if we cannot trust LLM outputs to be truthful, LLMs are very limited when asked to make predictions or forecasts. Even when truth is deducible from available data using logic, LLMs cannot reliably get to the truth.

This puts a large burden of responsibility on the user. Decision making often carries a hidden appraisal of the future. Users must treat the LLM as an error-prone tool and carefully evaluate its responses before making decisions based on them.

Plain Old ANNs

Artificial neural networks (ANNs) have the potential to be reliable forecasters. An ANN is a series of layers of neurons. The connections carry weighted outputs of a layer as the inputs of the next. Multilayered ANNs are often called deep learning networks. Every LLM relies on an embedded ANN as the inference engine that computes the response to a prompt. It uses the prompt to produce a most probable next word, then cycles that word back into the prompt. This embedding is called a "recurrent neural network." It tends to amplify low-probability outcomes and channel the LLM into hallucinations.

ANNs used on their own (no feedback) are different. A "universal approximation theorem" says that a sufficiently large ANN can approximate a bounded continuous function arbitrarily closely. That means there is an error bound E such that the output is within E of the correct output, where E is smaller for larger networks (more neurons and connections). The network is trained with a large set of (x,y) pairs sampled from the function. After training, the ANN implements an approximation g for the function where $|f(x)-g(x)| < E$. This works even if the input x is not in the training set. Continuous bounded functions are common as mathematical models of many physical processes. These models can be validated as shown the accompanying figure.

Unfortunately, some physical processes are not continuous bounded functions. The equations for weather prediction allow for local chaotic events, whose effects then spread throughout the model's grid. Training an ANN from samples of weather data will be confounded by the unboundedness of those events. ANN near-future weather predictions are more likely to be accurate than predictions for many days.

Other processes are discontinuous. For example, showing an ANN many samples of (image, label) will cause it to become an approximation for a function that maps images to labels. But a small change in input, such as modifying a few pixels of an image, can cause the model to output the label of a completely different image. This is called fragility: the output cannot be guaranteed to be anywhere close to the proper label when the network is presented with an image not in the training set. It is difficult to validate such models according to the scheme of the figure.

Non-LLM ANNs have been successful for short-term weather prediction, reinforcement learned games (chess, Go), protein folding, and more.

Conclusion

It seems unlikely that LLMs can be sufficiently well validated to be reliable predictors. Plain ANNs without feedback that have learned from samples of continuous bounded functions can be validated and reliable. Users seeking reliable predictions are advised to employ plain ANNs when possible. LLM users must independently verify the models before making decisions based on them. Users of unverified models will still be held responsible for their actions based on model outputs. Let us not be fooled by LLMs that interact with us in natural language—they are text generators with no intelligence or judgment behind them.

Footnotes

a See B. Schneier. "AI and trust." *Comm. ACM* 68, 8 (Aug. 2025), 29–31.

b See G. Marcus and E. Davis. *Rebooting AI: Building AI We Can Trust*. Vintage (2019).

c See J. Buzen. *Rethinking Randomness: A New Foundation for Stochastic Modeling*. CreateSpace (2015).

About the Authors

Peter J. Denning (pjd@nps.edu) is Distinguished Professor of Computer Science at the Naval Postgraduate School in Monterey, CA, USA, is Editor of *ACM Ubiquity*, and is a past president of ACM. His most recent book is *Navigating a Restless Sea: Mobilizing Innovation in Your Community* (with Todd Lyons, Waterside Productions, 2024). The author's views expressed here are not necessarily those of his employer or the U.S. federal government.

Join the Discussion (0)