Federal Register Notice: 89 FR 51554, Federal Register :: Networking and Information Technology Research and Development Request for Information on Digital Twins Research and Development, June 18, 2024.

Request for Information on the National Digital Twins R&D Strategic Plan

Joshua Zev Levin, Ph.D.

Chief Creative Officer, LeviCar Unlimited

Josh-Levin@LeviCar.com

Interactive Digital-Twins Generator using Artificial Intelligence

by Joshua Zev Levin, Ph.D.
Chief Creative Officer, LeviCar Unlimited
[redacted]

This paper describes a concept for an Interactive-Environment Program Suite to utilize Artificial Intelligence (AI) to generate Digital Twins. This is in response to the request for comments as specified in an RFI (Request for Information) from the National Science Foundation.[1]

Topic Keywords: **Artificial Intelligence (AI)**, **Data**, **Trustworthy**, **LLMs (Large Language Models)**, **SMLs (Small Language Models)**

AI Front End

Assume we have several companies that don't want to share their secrets. Each company has one or more teams of engineers, that may or may not want to share secrets. Each team may work on one or more projects. Some of the projects will use AI and start with LLMs, and the following discussion pertains to such projects, and what the "Generator" would do for them. Other projects that don't use AI are not affected by this discussion.

As each team of engineers commences each project, they would use Large Language Models (LLMs) to fish for ideas and concepts, and then refine things using custom Small Language Models (SLMs). This would be done according to the article by Kane Simms.[2]

It is desirable to make a custom LLM based on juried material, mostly derived from what is already on-line on the internet, but also peer-reviewed journals and respected books. The Chatbot might be based on the Proximity[3] chatbot, either by trying to reverse engineer Proximity, or licensing it from its owners. Proximity lists source materials, which is very useful to engineers. Other chatbots might be used, instead.

The database would be *juried*, that is, selected by professionals for accuracy. The database could be privately-owned and users would pay a fee, or it could be government-owned. In the latter case, I suggest that the National Institute of Standards and Technology (NIST) own and run it, as NIST already has considerable expertise with AI. This is in-line with the tradition of government aiding developing technologies.

Also, the owners of the LLM would also take care of the IP rights, compensating the owners of the source material for the use of said source material. Bear in mind that, here in the United States, the rationale for granting so-called “Intellectual-Property” (IP) rights is based on “to promote … useful arts …” [4], and any attempt to obstruct that would exceed the authority that Congress has the power to give. However, rights holders are entitled to some reasonable compensation.

According to Simms[2] once the engineers have reached the limits of what they can do with the LLM, they need to switch to a customized SML. He states specifically:

> “Once you’ve proven that the task is doable, you can then start working down in model sizes to figure out whether the same task can be done using a smaller model. When you reach a model size where your results start to change, get less accurate or slightly more unpredictable, you’ve reached your potential model size.”

He also states:

> “With LLMs, you’re in the hands of the model builders. If the model changes, you’ll have drift or worse, catastrophic forgetting. With SLMs, anyone can literally run them on your own servers, tune them, then freeze them in time, so that they never change.”

I myself disagree with the last phrase, “… then freeze them in time, so that they never change.” The main purpose of DT is to allow changes, and then virtually test them before committing them to extensive hardware or software development investments.

Each team could implement SMLs on their own firewalled servers, or the operators of the LLMs could established firewalled application within the LLMs to perform this. The question of which to use is beyond my own knowledge of computer security.

In the case of several SMLs coexisting on one LLM server, it is possible to have a supervisor application that can look beyond the firewalls, and, using AI, determine if there is a possibility of two or more teams coöperating. It is an open question as to how it would inform the teams that they have commonality, and for what they guidelines should be for sharing such information.

It is also possible to have different teams have prior arrangements to share some (but maybe not all) of their private data.

DT Back End

You might ask why, if the Front End and the Back End have two different functions, why not simply use two separate programs, instead of have one program suite?

The first reason is that the output of the Front End and the input of the Back End should use exactly the same format, so there is no miscommunication, and no need to have yet another program to "translate" the Front End's output into the Back End's input. Conventionally, once the AI Front End has been used to establish the engineering goals, then the Back End can be used to seamlessly model the finished product, giving the engineering teams the opportunity to "edit" the result and see if such modifications improve performance, longevity, efficiency, costs, and other factors, as outlined below. Once real models are built, measurements from those can be included in the DT, and this iterative enrichment of the DT content can result in a better product, which, after all, is the reason to use and maintain the DT.

However, there is yet a second, more-subtle, reason to keep the two functions together in one overall program suite. The Front End establishes engineering goals (AI, LMs), and the Back End provides a way to realize these goals. However, sometimes the Front End will miss something, and, which may be due to human error, or just simply the complexity of the particular project. Sometimes this might not be apparent until the Back-End output is examined. Perhaps the Digital Twin shows that the project does not (yet) live up to its goals. This might be detected by the engineering team (humans), or might be detected automatically by AI.

In either case, we can have recommendations on how to change the initial input parameters to the Front End. Perhaps the original questions were wrong. Perhaps a new tool or new technology has become available. By having a unified program suite, the two halves can work together to indicate what changes might be needed to the initial input. In other words, not only would the engineers have opportunities to change things and iterate within the Front End and within the Back End, they'd also have the opportunity to iterate between both parts, using errors detected at the output of the Back End to refine the inputs to the Front End.

Getting back to the functionality of the DT Back End: the Digital Twin of an engineered product can be used to make predictions, even before it is built or deployed, about its initial cost, performance, longevity, efficiency, continuing costs, ecological impact, human interface, and even human reactions to it. After the initial design, DTs can be use to continually model each instance of the product, as it is serviced and deployed. This can be aided by feedback from onboard sensors from each instance back to the DT. Details are in the Wikipedia article.[5]

Since DTs can include post-production data from servicing records and onboard sensors, feedback can be used within the Back End, and even between the Back End and the Front End, based upon how the engineered product works in the real world.

Conclusion

The suggested Program Suite can carry a proposed engineering product from the initial idea through several layers of design and implementation by harnessing Artificial Intelligence (AI) in the form of Large and Small Language Models (LLMs and SLMs), and by utilizing the power of Digital Twins (DTs), with feedback possible between different layers. Even post-deployment, feedback from servicing and sensors can lead the Program Suite to suggest modifications based on real-world experience.

The Program Suite can mitigate some of the problems associated with AI, such as using juried source material to reduce "hallucinations", and compensate owners of source material for their use.

References

(1) *National Science Foundation: Networking and Information Technology Research and Development Request for Information on Digital Twins Research and Development*. Federal Register / Vol. 89, No. 118 / Tuesday, June 18, 2024 / Notices pp. 51554-51555

(2) "10 differences between small language models (SLM) and large language models (LLMs) for enterprise AI" by Kane Simms (https://www.linkedin.com/pulse/10-differences-between-small-language-models-slm-large-kane-simms-edvee/) paywalled.

(3) Proximity Chatbot: Website: https://www.perplexity.ai/;
Wikipedia: https://en.wikipedia.org/wiki/Perplexity.ai

(4) U.S. Constitution, Article I, Section 8, Clause 8, "To promote the progress of science and useful arts, by securing for limited times to authors and inventors the exclusive right to their respective writings and discoveries."

(5) Wikipedia article: https://en.wikipedia.org/wiki/Digital_twin