Large Language Models in High Energy Physics (succinct survey) and directions of future developments

A. Shevel[a], A. Oreshkin[a], A.Shvetsov[a], A. Naikov[a]

[a] Petersburg Nuclear Physics Institute named by B.P.Konstantinov of NRC «Kurchatov Institute»

Integrating Large Language Models (LLMs) into high-energy physics (HEP) drives a paradigm shift in how researchers design experiments, analyze data, and automate complex workflows. A survey reveals various scenarios in which LLMs utilize pre-trained models through the Retrieval-Augmented Generation (RAG) architecture to meet their requirements. It analyzes our experimental scenario to show how the RAG architecture helps improve an administrative manual for the segment of sever network. Finally, it describes possible directions for future developments.

INTRODUCTION

LLMs, as part of the Machine Learning (ML) area, are in wide use everywhere now, including many aspects of High Energy Physics (HEP) experiments. It is helpful to recall that, among several concrete examples, the application of LLM and a more advanced version of LLM is Large Reasoning Models (LRM). The *reasoning* in LLM means that the model might explain the course of generating the answer or continuation of the entered text. Further, it is planned to use LLM for both versions. If LLM has been trained on an appropriate corpus of texts, LLM can generate consistent continuation.

LLM is trained with a large (many hundreds of millions of pages) corpus of texts that was collected (gathered) by the developers. Of course, the LLM training required a lot of computing power. It is difficult to estimate the computing power used in each case; however, the biggest LLM developers, such as OpenAI and Microsoft, have bought hundreds of thousands of GPUs. Many LLMs are proprietary. At the same time, the situation in the LLM landscape is changing almost every month, if not faster. As a result, in most cases, there is an opportunity to use an open-source, free-of-charge LLM instance.

There are examined examples (scenarios) of using LLMs in scientific laboratories. In [1], the complex experimental facility and advanced instrument upgrades are described. It is suggested that LLMs can help perform complex information retrieval, assist in knowledge-intensive tasks across applications, and provide guidance on tool usage. The authors describe preliminary experiments with a Context-Aware Language Model for Science (CALMS) to help scientists with instrument operations and complex experimentation. They use the Retrieval Augmented Generation (RAG) architecture with LLMs available on the Internet. They estimated that LLM usage could improve by up to a factor of 5. In [2], the authors propose using large language models (LLMs) to tune particle accelerators. The authors demonstrate a proof-of-concept example of the ability of open-source LLMs to tune an accelerator subsystem based solely on a natural language prompt from the operator. At the same time, they found that for a particle accelerator environment, traditional algorithms for accelerator tuning are cheaper and faster than using LLMs. They also show that LLM inferences depend on many details: type of prompts, type of LLM, etc. Another example of using an LLM (or scenario) in accelerators is in [3]. Authors explore jointly testing a tailored Retrieval Augmented Generation (RAG) model to enhance the usability of particle accelerator logbooks at institutes like DESY, BESSY, Fermilab, BNL, SLAC, LBNL, and CERN. They aim to use data from logbooks by leveraging their information content to streamline daily use,

enable macro-analysis for root cause analysis, and facilitate automation in problem-solving.

In [5], the authors introduce a developed model named HEP-Xiwu, trained on HEP data. This reasonably reduces hallucinations, although it may use several LLMs such as GPT-4, LLaMA, and Vicuna. In the suggested architecture, nearly any LLM can be used.

There are also examples of LLM applications in computing infrastructures [6,7]. Popular applications of LLM include searching various documents distributed across Intranet sites and local databases [8,9].

Another scenario specific to the HPC scientific research environment is discussed in [10]. The authors examined various scientific workflows in HPC and proposed a framework to test the model's effectiveness and performance in real-world conditions. They highlighted the importance of applying the FAIR (Findability, Accessibility, Interoperability, Reusability) principles within HPC-FAIR, which is dedicated to centralizing HPC-related datasets and AI models in a single hub.

The interesting scenario of LLM application is discussed in [11- 13], where automatic theorem proving is demonstrated if the theorem is formulated in an appropriate language, e.g., Lean 4.

Finally, there is a more general approach for future LLM architecture in the form of a fundamental Large Physics Model (LPM) [14]. The author's team suggested that LPM development might be achieved through interdisciplinary collaboration among experts in physics, computer science, and philosophy of science. To integrate these models effectively, they identify three key pillars: Development, Evaluation, and Philosophical Reflection. The other details are discussed as well.

The studies mentioned above do acknowledge the truth of the "No Free Lunch theorem" [15]. This theorem means in machine learning that no single model or learning algorithm is universally optimal for every dataset or task. Choosing an appropriate model often involves considering the nature of the data, the complexity of the task, and other contextual factors.

Let us emphasize that the consistency of the inference generated by LLM must be certified, i.e., a scientist needs to find a way to certify the truth of the inference from the LLM. The certification degree of an inference might be improved with the Retrieval Augmented Generation (RAG) architecture [16]. In such architecture, LLM should perform the "retrieval" process only in a curated corpus of texts. The additional approach might include a special tool described in [17].

The decision has remained on how a specific scenario fits the application environment. In other words, the effect of using LLM in a concrete scenario needs to be shown, that the usage of LLM gives better (e.g., unique or faster than before) results in comparison with traditional methods.

Now, examine the computing infrastructure in the HEP environment. Developing and maintaining computing systems is a complex endeavor due to the constant evolution of technology, changing requirements during development, and ongoing system maintenance throughout their lifetime. The high volume of maintenance work after deployment includes troubleshooting, patching, updating, and modifying components to accommodate new features or security requirements. Investigating unusual events or planning modernization may involve scanning system descriptions, administrative records, system logs, and other data sources – a truly multidimensional task. In these circumstances, it is worthwhile to consider using LLM architecture.

THE EXAMPLE SCENARIO of LLM USE IN COMPUTING INFRASTRUCTURE

The initial idea was to develop an artificial assistant (AA), where the developer/administrator enters a question in natural language and gets the answer in the same language. A simple chatbot prototype for interaction with the local description of the server network segment has been implemented. The architecture Retrieval Augmented Generation (RAG) [18] has been chosen (invented in ~2021) for this prototype.

Such architecture has significant **advantages**: the use of a **pre-trained LLM**, i.e., knowledge transfer; **local operations**, i.e., all retrieval and generation operations are performed on the local server; no local data is sent outside the LAN; and **local server resources** are sufficient for our local needs. The basic (naïve) version of RAG was in use.

Several setup versions were exploited: the Linux server Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz, 128GB, four simple GPUs "GeForce GTX 1080 Ti", python3.12, Linux; MAC book pro m3, 36 GB; multipurpose computing center at NRC "Kurchatov Institute" http://computing.nrcki.ru/ under SLURM with available for testing one GPU V100- PCIE-32GB. The initial Python program for RAG was obtained from https://gitlab.com/rahasak-labs/iollama (later redeveloped). Local LLM instances llama3.2, deepseek-r1:32b, aya:35b, qwen2.5:32b, etc., were used on platform ollama. The transformation of the text description into a vector database (or vector array) ("embedding") was used with several tools like LaBSE, paraphrase-multilingual-MiniLM-L12-v2, sbert_large_nlu_ru, etc., obtained from https://huggingface.co/. Web frontend was configured with the package streamlit from https://github.com/streamlit. The initial version of the description for the server network segment consisted of approximately 100 pages.

Initially, several questions were prepared and entered with the Web frontend. The output looked confusing: some answers were utterly wrong. Others were just fantasy (hallucination) – at first sight correct, but not true.

To certify the LLM, a review of the server network segment description has been done. The review revealed that several aspects of the description were unclear and incomplete as they should be. Our conclusion was: the description should be improved, the same question series must be entered, and the quality of the answers needs to be estimated again. Having made corrections, the developers of the description discussed many times "what is what and where it comes from" (very fruitful). Such a cycle was repeated several times.

The result: an improved version of the server network segment description (the content became more structured and complete, the terminology and the whole content have become more consistent, and the number of pages increased by about 50%). Examples of administrator interaction with the RAG setup are shown in [19]. Ultimately, we found out that the best LLM for our needs is deepseek-r1:32b, and the embedding model is ru-en-RoSBERTa. Side remark: the description is in Russian; however, if the entered question is in English, the answer will also be in English.

The RAG installation was used as the partner (or special peer reviewer, which is not easy to find, if it is possible at all) for developers to improve the document. Some practical considerations follow. A simple option for creating a test question list is to order it using one of the existing LLMs, such as deepseek-r1 or in your RAG. You can use a prompt like "Please prepare 70 test questions to check the quality of the server network description." The generated list of test questions might be entered into RAG one by one.

The quality of answers must be estimated on a scale of 0 to 10. The cycle might be considered as finished if the estimation mark is not less than 7 out of 10. Critical remark 1: The estimations are performed by the developers. The required time to produce the answer to a question was ~10 minutes in the current test prototype. Critical remark 2: In case of increasing document volume, e.g., by 10**3, a 10**3 times more powerful GPU cluster may be required. A description improvement time depends on several conditions: the proximity of the LLM to the description topics, the degree of conformity of the text-to-vector database transformation (embedding) correspondence to the description language/jargon, the volume of the description, and the available GPU, and the accuracy of instructions (prompt) to the LLM, choice of appropriate values of LLM parameters (temperature, top_p, num_ctx, etc), completeness and clarity of the initial version of the description, finally most important: qualification of the description developer's team.

This scenario's advantages in development or administration might simplify or reduce the following: the time needed to develop a description, the involvement of new participants in development or administration, the maintenance of the developed system, further modernization during operation, and discussions within the developer's team or administrators to clarify the RAG architecture, which might generate new ideas about a system being developed or already developed.

In the foreseeable future, the described technology will become a standard step in creating documentation for systems being developed. Further improvements include a full-scale digital twin (DT) of an extensive computing infrastructure. The DT is designed to help administrators minimize the effort required to resolve issues by considering not only the description but also all relevant logs, a set of software packages, manuals, hardware communication schemas, official rules, and orders, among other resources. Obviously, the RAG architecture in DT extends capabilities through deep contextual understanding of unstructured data. It can be expected that if DT+RAG development occurs in parallel with computing system development, it will enhance the computing architecture through a similar RAG cycle described above, much like text improvement.

CONCLUSIONS

Future LLM development and invention in a real HEP environment might be expected in the following directions:

- Full-scale digital twins, which include data search and analysis features for experimental facilities, e.g., accelerators, large detectors, and computing infrastructure.
- Fundamental LPM instead of relying on the general-purpose fundamental LLM.
- Architecture/protocols for data exchange between the above components and scientists.

REFERENCES

1. *Prince, M.H., Chan, H., Vriza, A. et al.* Opportunities for retrieval and tool-augmented large language models in scientific facilities. npj Computational Materials 10, 251 (2024) https://www.nature.com/articles/s41524-024-01423-2, https://doi.org/10.1038/s41524-024-01423-2 // p. 8.

2. *Jan Kaiser, Anne Lauscher, and Annika Eichler* "Large language models for human-machine collaborative particle accelerator tuning through natural language" // SCIENCE ADVANCES, 1 Jan 2025, Vol 11, Issue 1, https://www.science.org/doi/10.1126/sciadv.adr4173. // p. 13.
3. *Antonin Sulc et al.* "Towards Unlocking Insights from Logbooks Using AI" // https://arxiv.org/abs/2406.12881 [Submitted on 25 May 2024] // p. 4.
4. *Kelly B. Wagman, Matthew T. Dearing, Marshini Chetty //* Generative AI Uses and Risks for Knowledge Workers in a Science // https://arxiv.org/abs/2501.16577 [Submitted on 27 Jan 2025] // p. 19.
5. *Zhengde Zhang et al.* "Xiwu: A Basis Flexible and Learnable LLM for High Energy Physics" // https://arxiv.org/abs/2404.08001 [Submitted on 8 Apr 2024] // p. 15.
6. *Abane, A., Battou, A. and Merzouki, M.* (2024), "An Adaptable AI Assistant for Network Management", NOMS 2024-2024 IEEE Network Operations and Management Symposium, Seoul, KR, [online], https://doi.org/10.1109/NOMS59830.2024.10574957, https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=957878 (Accessed August 18, 2025) // p. 3.
7. *Nimesh Jha et al. "*LLM Assisted Anomaly Detection Service for Site Reliability Engineers: Enhancing Cloud Infrastructure Resilience" // https://arxiv.org/abs/2501.16744 [Submitted on 28 Jan 2025] // p. 9.
8. *Verena Kain et al. "*A CERN Knowledge Retrieval Chatbot (Harnessing the power of AI for efficient information retrieval)" [presentation at CHEP-2024 https://indico.cern.ch/event/1338689/contributions/6010661/] // p. 26.
9. *Dal Santo, Daniele* et al. "chATLAS: An AI Assistant for the ATLAS Collaboration" // https://cds.cern.ch/record/2935252# [Presentation.] CERN LHC ATLAS collaboration on 10 Jun 2025] // p. 31.
10. *Liao, Chunhua; Shen, Xipeng; Emani, Murali; Vanderbruggen, Tristan; Lin, Pei-Hung* "HPC-FAIR: A Framework Managing Data and AI Models for Analyzing and Optimizing Scientific Applications" // https://www.osti.gov/servlets/purl/2504172 // Technical Report OSTI ID 2504172 // 30 October 2024 // p. 27.
11. *Kefan Dong, Arvind Mahankali, Tengyu Ma* "Formal Theorem Proving by Rewarding LLMs to Decompose Proofs Hierarchically" // https://arxiv.org/abs/2411.01829 [Submitted on 4 Nov 2024] // p. 20.
12. Kefan Dong, Tengyu Ma "STP: Self-play LLM Theorem Provers with Iterative Conjecturing and Proving" // https://arxiv.org/abs/2502.00212 [Submitted on 31 Jan 2025 (v1), last revised 21 Mar 2025 (this version, v4)] // p. 25.
13. *Leni Aniva, Chuyue Sun, Brando Miranda, Clark Barrett, Sanmi Koyejo* "Pantograph: A Machine-to-Machine Interaction Interface for Advanced Theorem Proving, High Level Reasoning, and Data Extraction in Lean 4" // https://arxiv.org/abs/2410.16429 [Submitted on 21 Oct 2024 (v1), last revised 31 Jan 2025 (this version, v2)] // p. 19.

14. *Kristian G. Barman et al.* “Large Physics Models: Towards a collaborative approach with Large Language Models and Foundation Models” // https://arxiv.org/abs/2501.05382 [Submitted on 9 Jan 2025] // p. 27.
15. *D. H. Wolpert and W. G. Macready*, "Coevolutionary free lunches," in IEEE Transactions on Evolutionary Computation, vol. 9, no. 6, pp. 721-735, Dec. 2005, doi: 10.1109/TEVC.2005.856205 // https://ieeexplore.ieee.org/document/1545946.
16. *Patrick Lewis et al.* “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks” // https://arxiv.org/abs/2005.11401 // [Submitted on 22 May 2020 (v1), last revised 12 Apr 2021] // p. 19.
17. Vinay Kumar Sankarapu et al. “DLBacktrace: A Model Agnostic Explainability for any Deep Learning Models” // https://arxiv.org/abs/2411.12643 [Submitted on 19 Nov 2024 (v1), last revised 4 Feb 2025 (this version, v2)] // p. 20.
18. Shailja Gupta, Rajesh Ranjan, Surya Narayan Singh “A Comprehensive Survey of Retrieval-Augmented Generation (RAG): Evolution, Current Landscape and Future Directions” // https://arxiv.org/abs/2410.12837 [Submitted on 3 Oct 2024] // p. 18.
19. Alexey Naikov, Anatoly Oreshkin, Alexey Shvetsov, Andrey Shevel “The machine learning platform for developers of large systems” https://arxiv.org/abs/2501.13881 [Submitted on 23 Jan 2025 (v1), last revised 20 May 2025 (this version, v4)] // p. 12.