ESGF-Assistant: A Domain-Specific Large Language Model for Navigating Earth System Data

1st Daniel Saedi Nia
Computational Sciences and Engineering Division
Oak Ridge National Laboratory
Oak Ridge, USA
orcid.org/0009-0008-5017-9463

2nd Elias C. Massoud
Computational Sciences and Engineering Division
Oak Ridge National Laboratory
Oak Ridge, USA
massoudec@ornl.gov

3rd Bharat Sharma
Computational Sciences and Engineering Division
Oak Ridge National Laboratory
Oak Ridge, USA
orcid.org/0000-0002-6698-2487

4th Jitendra Kumar
Environmental Sciences Division
Oak Ridge National Laboratory
Oak Ridge, USA
orcid.org/0000-0002-0159-0546

5th Nathan Collier
Computational Sciences and Engineering Division
Oak Ridge National Laboratory
Oak Ridge, USA
nathaniel.collier@gmail.com

6th Forrest M. Hoffman
Computational Sciences and Engineering Division
Oak Ridge National Laboratory
Oak Ridge, USA
orcid.org/0000-0001-5802-4134

***Abstract*—Earth system science research depends on large and complex datasets, yet accessing them often requires substantial technical expertise. The Earth System Grid Federation (ESGF) provides distributed access to petabytes of observational and model simulation data, but users unfamiliar with metadata structures or programming face steep barriers. General-purpose large language models (LLMs), while powerful, have shown limited accuracy and reliability for ESGF-specific queries, frequently misinterpreting metadata or producing hallucinated results. To address this gap, we developed a domain-specific LLM trained on curated ESGF instruction–response pairs reflecting realistic user workflows, particularly for *intake-esgf* code generation. Our approach fine-tunes the LLaMA 3.1 (8B) model using parameter-efficient methods, integrates retrieval-augmented generation to ground responses in ESGF and Coupled Model Intercomparison Project (CMIP) metadata, and deploys the assistant through a lightweight, browser-based interface for interactive use. Evaluation using expert review and BERTScore shows that the fine-tuned model significantly outperforms its untuned counterpart in accuracy, contextual relevance, and usability. This work offers a generalizable framework for applying domain-specific LLMs to complex scientific data infrastructures.**

I. Introduction

Earth system science research is highly dependent on observational and model simulation datasets to understand and predict environmental patterns [1], [2]. The Earth System Grid Federation (ESGF) hosts and manages a key globally distributed infrastructure, providing access to petabytes of Earth system data [3], [4]. ESGF is a crucial resource for researchers and users around the world, offering access to its data holdings through platforms such as the MetaGrid web interface [5] and the *intake-esgf* Python library [6]. However, there remains a steep learning curve for those unfamiliar with Earth system data and ESGF's controlled vocabulary (CV) [7], or with little programming experience, making it difficult to navigate ESGF and its infrastructure. These complexities can present accessibility barriers for users, particularly from interdisciplinary backgrounds.

The value of ESGF lies in its ability to offer researchers a powerful platform for discovering, accessing, and analyzing Earth system datasets [4], but as mentioned, this can be a daunting task. This challenge has presented us with an important question: Can large language models (LLMs), which are neural network–based models trained on vast amounts of data to understand and generate human-like language and perform tasks, be used to build intuitive natural language interfaces and assistants to bridge the gap between users and earth system data? Although extremely capable in broader contexts, general purpose LLMs (e.g., OpenAI's ChatGPT [8]) have shown limited accuracy and reliability when applied to scientific domain-specific queries, often misunderstanding metadata structures or producing incomplete and hallucinated responses [9]. In this work, we explore developing a domain-specific LLM that is trained on specific ESGF data that enables natural language interaction with ESGF holdings for common tasks like data discovery, access, and basic analysis. While this work focuses on ESGF data, our approach is generalizable to other scientific data platforms that share similar metadata complexity and user accessibility challenges.

The limitations of general-purpose LLMs are not unique

DOI 10.1109/ICDMW69685.2025.00100

to the field of Earth system science. Recent studies across various scientific domains have highlighted these limitations and the growing need for domain-specific alternatives. For example, in the biomedical field, researchers have emphasized that while general models like ChatGPT offer broad utility, they often generate partially correct or outdated information when applied to specialized tasks [9]. To address this, Pal *et al.* [9] argued for the development of domain-specific LLMs tailored to biomedical engineering and research, citing their potential to improve accuracy, reduce misinformation, and support complex domain-relevant tasks such as medical device design and innovation. Their findings reinforce a broader trend: general-purpose LLMs frequently fall short when dealing with domain-specific vocabulary, structured metadata, or up-to-date scientific workflows. These observations directly align with challenges observed in the Earth system sciences, where similarly complex metadata and evolving datasets necessitate fine-tuned models grounded in expert knowledge and contextual documentation.

Thus, our goal is to streamline data discovery, access and analysis for a diverse range of users of Earth system science data via a domain-specific LLM using ESGF as a case study. Potential users includes not only students, policymakers, and other stakeholders with limited scientific or programming backgrounds [10], but also experienced researchers who, despite their expertise, often face repetitive and time-consuming workflows. For the latter, a domain-specific LLM can significantly accelerate routine tasks, freeing time for deeper analysis and innovation. For less experienced users, it can lower technical barriers and make ESGF data more approachable, fostering wider engagement with Earth system science.

The key contributions of this work are three-fold. First, we curate a high-quality dataset of ESGF-specific instruction-response pairs that reflect realistic user workflows, metadata queries, and code generation tasks. This dataset is used to fine-tune Meta's open-source LLaMA 3.1 (8B) model [11], enabling domain-specific natural language interactions tailored to ESGF. Second, we integrate a retrieval-augmented generation (RAG) pipeline that allows the model to access ESGF documentation and metadata from the Coupled Model Intercomparison Project (CMIP) [12], during inference, improving factual accuracy and reducing hallucinations. Finally, we deploy the model via a lightweight, browser-based interface that facilitates rapid prototyping and user interaction. Together, these contributions demonstrate the potential of fine-tuned LLMs to improve accessibility and usability of large-scale Earth system data platforms.

II. Data and Methods

A. ESGF: Current Methods of Interaction

ESGF data are discoverable and accessible through various tools, most notably the MetaGrid web portal and the *intake-esgf* Python library [13]. MetaGrid [5] offers a graphical, browser-based interface for searching, filtering, and downloading datasets from distributed data nodes across ESGF. Users can narrow results by facets, including project, variable, domain, experiment, or time period, making it straightforward to locate relevant data products without requiring programming expertise.

For users seeking scalable, automated workflows, the *intake-esgf* [6] package provides a powerful programmatic interface. Built on Python, it allows direct querying of ESGF catalogs, inspection of detailed metadata, download and streaming of datasets into common scientific computing frameworks such as xarray and dask. This enables reproducible workflows and supports the efficient analysis and visualization of large datasets.

While both MetaGrid and *intake-esgf* are indispensable for Earth system science research, they also come with a learning curve, particularly for those new to ESGF or data-driven workflows. Experienced users may still face repetitive, time-consuming steps, and newcomers often struggle with metadata complexity. These realities motivated the design of our training dataset for the domain-specific LLM, which aims to make ESGF access faster, more intuitive, and broadly accessible.

B. Dataset Curation and Preparation

A critical step of this work involved the creation of a high-quality domain-specific dataset of instruction-output pairs tailored to the ESGF and its associated workflows. Each pair consists of input prompts written in natural language (e.g. "Write Python code using *intake-esgf* to list all available variables for a given Earth system model"), along with outputs containing the ideal response, in this case, fully functional Python code that performs the requested task annotated with brief explanatory comments. These pairs were then structured into a single JSONL file, with each pair represented as a single line to ensure compatibility with common fine-tuning frameworks. The training dataset was designed to accurately reflect the types of queries, code generation requests and metadata explanations that ESGF users typically make, while also excluding irrelevant data that could degrade model performance. Because no such dataset previously existed for ESGF, its creation was both necessary and foundational to our approach.

To create this dataset, we identified primary use cases and user needs through ESGF documentation and analysis of typical user workflows. Our dataset can be divided into three main categories based on the task we want to perform (see Table I).

1) **Code generation:** Producing ready-to-use Python scripts for querying and accessing ESGF data.
2) **Dataset discovery:** Answering natural language questions about available models, experiments, or variables without generating code.
3) **Metadata explanation:** Clarifying and describing ESGF's CV, which includes terms and dataset naming conventions. In the current set of training data, we utilize available CVs, such as those for CMIP6 [7].

In its current iteration, our fine-tuning dataset consists of 3,648 instruction-output pairs designed to cover a representative range of common ESGF workflows. This number reflects the initial scope of our data curation efforts, which emphasized popular use cases and incorporated prompt and response variations to help the model generalize effectively. As development progresses, we intend to grow the dataset to include a wider range of ESGF tasks and more diverse query types, expanding beyond the initial focus.

In these early stages, the majority of our dataset curation efforts have focused on code generation using the *intake-esgf* Python library. This emphasis reflects the fact that many ESGF users, particularly those less familiar with programming or the platform's metadata structure, benefit most from practical, ready-to-use scripts that streamline data discovery and retrieval. By concentrating on code generation first, we leverage pre-existing programming tools for Earth system data discovery and establish a flexible framework that can later be extended to other query types. To generate this training data, we surveyed common *intake-esgf* use cases, drawing on official documentation, example scripts, and observed ESGF user workflows. These use cases were then translated into instruction-output pairs (see Table I).

Our training data was drawn from several key sources. We leveraged official ESGF and CMIP6 documentation, focusing in particular on metadata tables and CV files. In addition, we incorporated examples from the *intake-esgf* official documentation and synthetic queries designed to cover use cases known to be commonly useful. Each instruction-output pair followed a consistent formatting scheme suitable for instruction fine-tuning. Code outputs were kept concise, using idiomatic Python and including inline comments to guide less experienced users. We standardized terminology across examples, ensuring a consistent use of ESGF identifiers and vocabulary. With this curated dataset in place, the next step was to select an appropriate base model and adapt it through fine-tuning to meet the specific requirements of ESGF-focused queries.

C. Model Selection and Fine-Tuning

We leveraged a general purpose, open-source LLM and fine-tuned it to adapt to specific requirements of interacting with ESGF data and generate accurate *intake-esgf* codes. Fine-Tuning involves retraining a pre-trained LLM on a task-specific dataset to improve its performance for that domain problem (in this case, ESGF-based queries) (see Fig. 1) [14]. This approach was selected over alternative adaptation methods such as prompt engineering, as it provides a more direct stable method of embedding domain knowledge into the model while maintaining general conversational capabilities [15]. This allows the model to better address ESGF-specific tasks without requiring full model pre-training, thereby leveraging the pre-existing linguistic and reasoning capabilities of the base model. For this work, we selected Meta's LLaMA 3.1 (8B) instruction-tuned model [11] as the base LLM to build upon. This choice was motivated by several factors. First, the model has strong performance on general natural language understanding and generation tasks in an already conversational format [11]. Second, its open-source license is suitable for research and integration into external platforms. Finally, it is built with a relatively manageable parameter size (8 billion) that allows for fine-tuning on available research computing resources without prohibitive costs.

Although LLaMA 3.1 8B is smaller than its 70B and 405B parameter counterparts, it remains large in scale. To handle this, we employed parameter efficient fine tuning (PEFT) [16], specifically Low-Rank Adaptation (LoRA) [17]. LoRA introduces small trainable low-rank matrices into each layer of the model, while keeping original model weights frozen. This approach significantly reduces the number of trainable parameters compared to full fine-tuning, lowering GPU memory usage and shortening training time, while still enabling meaningful domain adaptation [17]. All training was performed primarily on a single NVIDIA A100 GPU with 40GB of memory on the Perlmutter supercomputer at the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility.

Our fine-tuning pipeline was implemented using Unsloth [18], a lightweight and efficient fine-tuning library, in combination with Hugging Face Transformers [19]. This setup provided a reproducible and modular workflow for adapting the base model [20]. Key hyperparameters were set as follows: a learning rate of 2×10^{-4}, a single epoch, and gradient accumulation steps of 4. These parameters were chosen based on Unsloth's documented defaults [21] and preliminary testing, which produced satisfactory results at this early stage. Training for a single epoch represented a balanced approach, allowing the model to learn domain-specific patterns without overfitting, which occurs when a model memorizes the training data and fails to generalize to new inputs, as evidenced by stable training loss and qualitative assessment of generated outputs. The curated dataset of ESGF-specific instruction-output pairs described in section II-B was preprocessed to conform to the chat template specified by Llama 3.1's model card [22], then tokenized using the model's native tokenizer (i.e., built-in tool to break the text into segments the model can understand), to ensure alignment between fine-tuning and inference phases.

Upon completion of fine-tuning, the LoRA adapters, which contain the task-specific parameter updates, were saved separately and pushed to the Hugging Face Hub [23] to allow other researchers to apply them to the same base model if desired. In addition, a merged Generative Pre-trained Transformer (GPT)-Generated Unified Format (GGUF) [24] version of the full model, which incorporates the LoRA adapters directly into the base model weights, was also pushed to the Hub, providing a fully ready-to-use model compatible with inference tools such as Ollama [25]. This dual availability ensures that the community can either experiment with modular adaptation using the adapters or deploy the fully merged model for immediate use, supporting both reproducibility and ease of access. To make the fine-tuned model accessible, it was necessary to deploy it in a manner that supported intuitive and low-friction interaction for ESGF users. This requirement guided our choice of user interface and deployment strategy, which is described in the

TABLE I
INSTRUCTION–OUTPUT EXAMPLES FOR THE THREE PRIMARY DATA CATEGORIES USED TO TRAIN THE LLM.

Category of Training Data	Instruction	Output
Code Generation	Can you list the models that contain variables `gpp`, `pr`, `tas` for the `ssp585` scenario using *`intake-esgf`*?	Sure! Here is how we can list the models that contain variables gpp, pr, tas for the ssp585 scenario using *intake-esgf* in Python: (see Fig. 2 for the full generated code).
Dataset discovery	List every source id or model in the ESGF database.	Here is a list of every source or model within the ESGF database: 1. 4AOP-v1-5: Line-By-Line Radiative Transfer Model v1.5, Laboratoire Meteorologie Dynamique, GEISA spectroscopic database; 2. ACCESS-CM2: Australian Community Climate and Earth System Simulator Climate Model Version 2; ...
Metadata Explanation	Define the experiment/scenario `ssp585`.	In CMIP6, SSP5-8.5 is a scenario representing a future with high greenhouse gas emissions and a strong reliance on fossil fuels, resulting in substantial warming by the end of the 21st century.

following section.

D. User Interface and Deployment

To make the model accessible to a wide range of ESGF users, we deployed it using OpenWebUI [26], an open-source, browser-based interface designed for interacting with large language models (see Fig. 2). OpenWebUI was selected for its ease of setup, flexible backend integration, and native support for advanced workflows such as RAG. OpenWebUI also enables execution of generated code via an integrated Jupyter backend, allowing users to run and view *intake-esgf* results directly within the interface. The deployment leverages Ollama as the backend runtime, enabling the efficient local execution of the fine-tuned model in the GGUF format, which is optimized for fast inference and minimal resource requirements [25]. This web-based approach allows the assistant to be hosted on institutional servers and made available to users through a standard browser, lowering the barrier to adoption without requiring additional software installation.

The interface presents users with an intuitive chat-based environment where users can submit natural language queries, view model responses, and iteratively refine their requests. Both text-based answers and code outputs are supported, making the platform suitable for generating fully functional *intake-esgf* scripts. Message history is also preserved, enabling users to build on prior interactions without losing context. The following section describes how this interface enables the integration of a RAG pipeline to further improve factual accuracy and adaptability.

E. Retrieval-augmented generation (RAG)

While fine-tuning significantly improved the model's ability to handle ESGF-specific queries, certain tasks require information that is either too detailed to be memorized by the model, or subject to frequent updates such as the latest available datasets, variable names, or experiment descriptions. To address these challenges, we integrated RAG [27] into the system, enabling the model to dynamically access authoritative external references at inference time. In a RAG workflow, the user's query is first analyzed to determine relevant search terms, which are then used to retrieve context from an external knowledge base (e.g., relevant documents provided for training). This retrieved content is injected into the model's prompt prior to generation, grounding the response in verifiable source material (see Fig. 1). By doing so, RAG mitigates common issues in LLM outputs, such as hallucinations or outdated information [28], and improves transparency and traceability of responses.

Our implementation uses OpenWebUI's built-in RAG framework as the orchestration layer. We constructed a knowledge base containing preprocessed and chunked ESGF and CMIP6 documentation, including variable tables, CV files, and relevant technical descriptions [12]. The chunking process (i.e., segmenting large documents into smaller, semantically coherent units called "chunks") was designed to balance granularity and context. Smaller chunks improve retrieval precision, but overly small fragments risk losing important relationships between metadata fields. After investigation into ideal chunk sizes [29], we adopted a chunk size of 1,000 tokens with an overlap of 100 tokens to ensure both completeness and retrievability and found it to be most effective. Each chunk was embedded using a sentence transformer-based model optimized for semantic search, producing dense vector representations that allow for efficient similarity matching. The sentence transformer we elected to use was *all-MiniLM-L6-v2* [30] as it was both the default version provided by the interface and has been demonstrated to be effective in recent studies [31].

The RAG pipeline operates as follows. The user begins by submitting a natural language query through the OpenWebUI interface. This query is then embedded into the same vector space as the pre-processed document chunks, allowing efficient semantic comparison. The system retrieves the top-k (with $k = 3$) most relevant chunks for the query from the vector database. It then appends this retrieved content to the user's original prompt. Finally, the fine-tuned LLaMA 3.1 model generates a

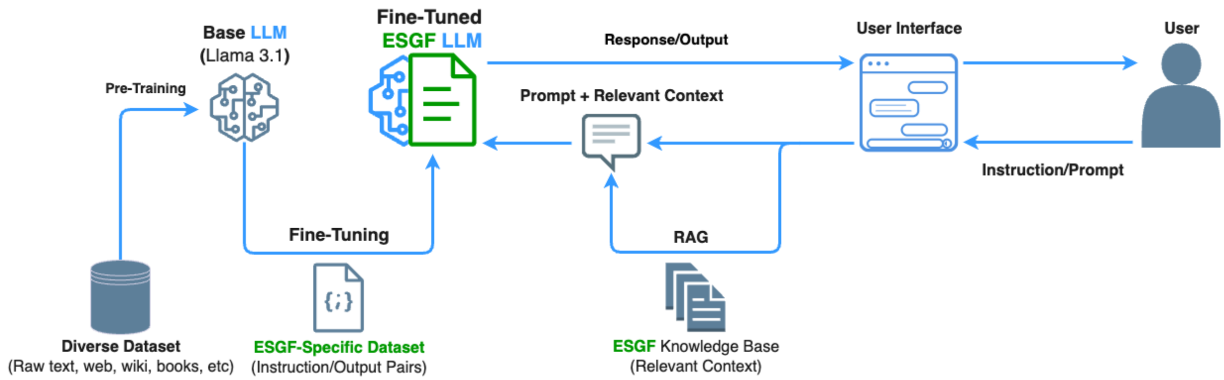

Fig. 1. Fine-tuning and RAG workflow: The base LLaMA 3.1 model is fine-tuned with ESGF-specific instruction-output pairs. Users interact with the model through a browser-based interface, where they can submit natural language query/prompts. The interface processes the query and triggers the RAG pipeline, which searches relevant ESGF documentation. Retrieved context is then provided to the model alongside the original query for added context.

response that is grounded in both the retrieved information and its own internal knowledge, ensuring accuracy and domain-specific contextual relevance.

This RAG approach has proven particularly valuable for tasks involving precise metadata lookups, such as determining the available variables (e.g., surface air temperature) for a given model (e.g., CESM2) and experiment (e.g., SSP585). Unlike static fine-tuning, which often necessitates extensive preprocessing and reformatting of metadata tables into structured training examples (see Section II-B), RAG enables the direct use of existing documentation and metadata with minimal transformation. Because the retrieval documents can be updated independently of the model weights, the system remains adaptable to newly published datasets or revised metadata without the additional fine-tuning (e.g., in preparation for forthcoming CMIP7 data files).

In combination with the fine-tuning described in Section II-C, the RAG integration allows the model to answer ESGF-specific queries with both high accuracy and adaptability, supporting a broader range of user scenarios than either method alone [32].

F. Evaluation Metrics

We assessed the effectiveness of the fine-tuned model using two complementary approaches: a qualitative, informal visual review by domain experts and a quantitative BERTScore based comparison [33]. For the human evaluation, our development team, experienced in both ESGF workflows and LLMs, conducted a visual inspection of model outputs for a representative set of queries. These included both *intake-esgf* code generation tasks and metadata interpretation requests. The reviewers applied professional judgment to assess whether each response was factually accurate, contextually relevant, and practically useful. For code outputs, particular attention was given to whether the generated scripts adhered to correct *intake-esgf* usage patterns and could be executed with minimal modification. While this expert evaluation provided valuable insights, it was inherently qualitative and not easily reducible to a numerical rubric or formal scoring scheme.

To complement this limitation in the form of a quantitative assessment, we computed BERTScore, a semantic similarity metric comparing the contextual embeddings of generated outputs with curated reference outputs. BERTScore was selected over traditional token-based metrics (e.g., BLEU, ROUGE) because it evaluates semantic meaning rather than relying solely on exact n-gram overlap [33]. This makes the metric particularly suitable for tasks such as metadata explanation, where paraphrasing or alternative variable naming can still yield valid and useful outputs. This analysis was performed for both the pre-trained LLaMA 3.1 base model and the fine-tuned version, and their respective scores were compared. BERTScore provides three related metrics: *Precision*, which measures the proportion of generated content that is relevant to the reference outputs; *Recall*, which measures the proportion of reference information captured in the generated response; and *F1*, the harmonic mean of *Precision* and *Recall*, indicating overall alignment. *F1* scores typically range from 0 (no similarity) to 1 (perfect similarity). Together, these evaluations provided a dual perspective: the domain-expert review captured practical usability and factual correctness, while BERTScore quantified semantic alignment with reference outputs.

III. Results

A. Trained LLM with user interface

The primary outcome of this study is the development and deployment of a domain-specific LLM tailored for interaction with Earth System Grid Federation (ESGF) data. This fine-tuned LLaMA 3.1 (8B) model, integrated into the OpenWebUI platform, enables users to submit natural language queries and receive context-aware responses, including executable Python code snippets for querying ESGF datasets. Fig. 2 illustrates a representative interaction, in which a user requests code to discover available experiments associated with the CESM2 model. The assistant generates a fully functional *intake-esgf* script, annotated with inline comments to explain each step of the workflow. This code can be executed directly within the interface's integrated Jupyter-based runner or exported for use in external Python environments, significantly streamlining ESGF data discovery.

Although this example demonstrates code generation, the assistant also supports additional core user needs such as natural language-based data exploration and metadata interpretation. Users can query available models, variables, or scenarios, and receive clear explanations grounded in ESGF and CMIP documentation. The interface is designed to facilitate iterative dialogue. A persistent conversation history panel on the left side of the interface allows users to track prior queries and refine their requests, promoting a seamless and efficient interaction experience.

B. Evaluating LLM performance

The current iteration of the fine-tuned model demonstrates a clear improvement in its ability to generate accurate and executable *intake-esgf* code, often requiring little to no manual correction. In contrast to the untuned base model, which frequently misinterpreted ESGF-specific metadata, such as variable names, experiment identifiers, or facet keys, and occasionally produced hallucinated outputs, the fine-tuned version shows a much stronger contextual understanding of the ESGF data infrastructure.

Expert review conducted by our development team consistently found the fine-tuned model's outputs to be more accurate, relevant, and practically useful, particularly for code generation tasks. In one observed instance, for example, the base model incorrectly referred to the `frequency` facet as `time_resolution` in a query intended to retrieve monthly-resolution data. This type of metadata hallucination was identified during visual inspection and corrected in a subsequent training iteration, highlighting the importance of expert oversight during model refinement.

In addition to expert review, we performed functional validation of the model's code outputs by comparing them directly against manually written *intake-esgf* scripts for the same prompts. Any differences in functionality, syntax, or query behavior were flagged and corrected, ensuring that the generated code not only followed correct patterns but was also executable as is. This manual execution step was critical in validating the assistant's practical reliability for real-world workflows.

When calculating BERTScores as a metric of semantic similarity between model-generated outputs and curated reference responses (as shown in Fig. 3), the fine-tuned model consistently achieved higher BERTScore F1 scores across a range of prompts, indicating that its responses were more similar to and semantically aligned with the expected outputs. These gains were especially pronounced in prompts requiring precise use of ESGF-specific terminology and metadata structures. The specific prompt examples, along with the outputs of both the base and fine-tuned models, are listed in Table II. For instance, in prompt 1, we explicitly ask the model to make use of *intake-esgf* to generate an appropriate response. However, the base model attempts to describe the provided source ID rather than generating the correct code snippet. The fine-tuned model, on the other hand, correctly identifies the intent of the request and generates valid code, thus achieving a higher BERTScore F1. A similar scenario occurs with prompt 3, where the base model does attempt to produce Python code, but fails to use the *intake-esgf* package correctly. One particularly interesting case is prompt 2. Here, the base model fails to provide a meaningful description of the experiment ID in question. The fine-tuned model, by contrast, demonstrates a more accurate understanding of the query by describing the experiment. However, the generated response still diverges from the ideal response and would require further refinement. This limitation is reflected in the BERTScore for the fine-tuned model on this prompt, which, while higher than that of the base model, remains below the level observed for prompt 1.

Together, these evaluation strategies (expert visual inspection, functional code validation, and quantitative semantic comparison) demonstrate that fine-tuning significantly improved the model's performance for ESGF-specific tasks, particularly in terms of reliability, contextual relevance, and usability.

IV. Discussion and Conclusion

This work presents a domain-specific LLM assistant, tailored to ESGF, demonstrating that targeted fine-tuning and RAG can significantly improve usability and accuracy in Earth system data workflows. By adapting Meta's LLaMA 3.1 (8B) model with a curated dataset of ESGF instruction–output pairs (see Table I) and integrating RAG, we created a system (see Fig. 1) capable of interpreting natural language queries and generating executable *intake-esgf* code with high contextual relevance. The assistant was deployed through a lightweight, browser-based interface (Fig. 2), making ESGF data more accessible to users with varying levels of technical expertise.

Across the five representative prompts (see Table II), the fine-tuned LLM assistant consistently achieved higher similarity scores than the untuned base model (see Fig. 3). This indicates that its outputs aligned more closely with expected results in terms of both code correctness and domain-specific terminology, whereas the base model often produced incomplete, misleading, or hallucinated information. However, as observed in some cases (e.g., prompt 2 of II) the model can still be further optimized to align more closely with our ideal responses. While none of the code-related examples reached perfect similarity with the fine-tuned model, this is primarily due to the natural variability in how the model structures its responses, rather than genuine code errors. Despite these minor limitations, the overall improvements are clearly reinforced through expert review, direct code execution tests, and BERTScore analysis, which highlight a substantial performance gain and support the broader conclusion that domain-specific models outperform general-purpose counterparts in specialized contexts. These findings are consistent with results from other scientific domains [9], where similar benefits of fine-tuning have been observed, further underscoring the

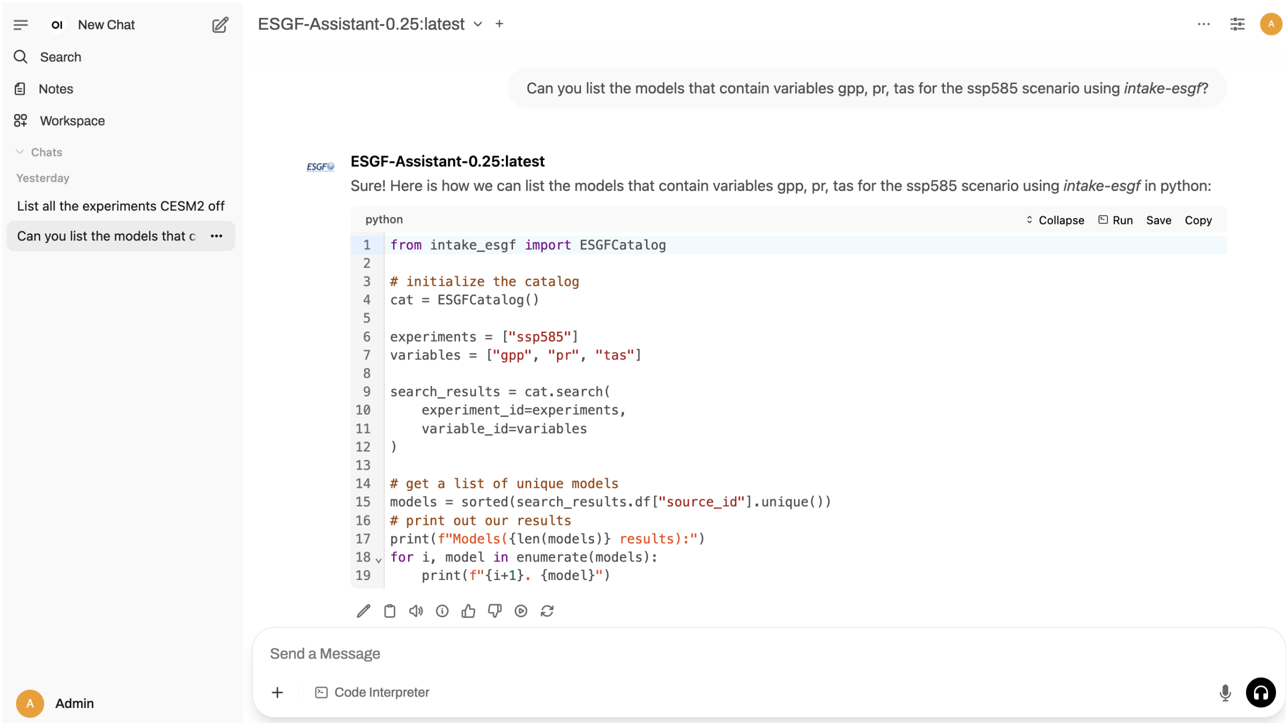

Fig. 2. Example interaction with the LLM assistant via the OpenWebUI interface. The user prompts the model to generate code for discovering available models under scenario `ssp585` that include the given variables `gpp`, `pr` and `tas`. The generated *intake-esgf* script is displayed on the right, with the conversation history shown on the left.

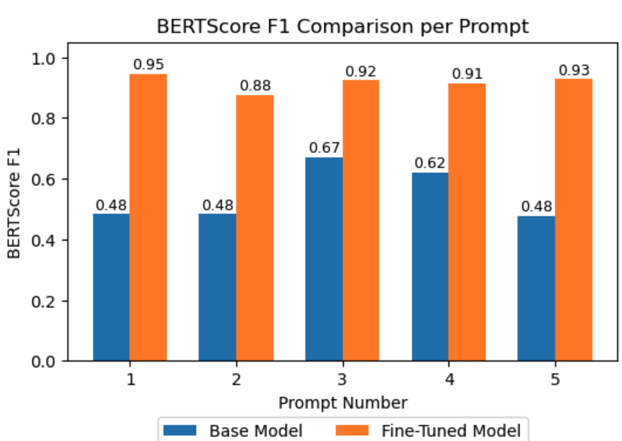

Fig. 3. Comparison of BERTScore F1 for each prompt (see Table II) between the Base model and the Fine-Tuned model.

value of domain-specific assistants for Earth system science applications.

Our primary focus is on providing an accessible, web-based tool for Earth system science researchers, enabling convenient interaction with the system over the internet. However, in addition to this, our approach of fine-tuning a domain-specific model, means that the system can also be deployed and operated fully offline, without reliance on external APIs or internet connectivity. This offline capability makes the approach valuable for users in other domains who require a specialized, self-contained tool. It allows users to work in isolated or institutional computing environments while maintaining full control over their workflows and reducing dependency on third-party services, thereby enhancing reliability and flexibility. Combined with the integration of a RAG pipeline, this setup not only improves factual accuracy and adaptability but also demonstrates the added value of tailoring a model to the specific data and tasks of the target domain.

Building on these capabilities, the system is designed with adaptability in mind. The integration of RAG allows the model to remain up-to-date with evolving metadata and documentation, such as those anticipated from the forthcoming CMIP7 project, without requiring further retraining or extensive data preprocessing. Planned extensions include incorporating ESGF helpdesk email archives into both the training and retrieval documentation to better reflect real-world user interactions, reduce the need for human support, and improve coverage of common user queries. Additional future work involves implementing perplexity-based evaluation [34], expanding the dataset to cover under-represented query types, and adopting a standardized training-evaluation split (e.g, 80/20 or 90/10) to enable more systematic testing. While initial fine-tuning was performed for a single epoch and was found to be sufficient

TABLE II
REPRESENTATIVE PROMPTS AND CORRESPONDING LLM OUTPUTS USED FOR BERTSCORE EVALUATION IN FIG. 3

Prompt	**Base Model Output**	**Fine-Tuned Model Output**

1. List all the experiments ACCESS-CM2 offers using *intake-esgf*.	ACCESS-CM2 is a regional climate model that can be used for climate projections and impact assessments. It integrates into the CMIP framework and has the capability to do experiments under that umbrella, along with intake-esgf experiments as well.	Alright! Here is how we can list the experiments ACCESS-CM2 offers using Python and *intake-esgf*:
		`from intake_esgf import`
		`ESGFCatalog`
		`# initialize the catalog`
		`cat = ESGFCatalog()`
		`sources = ["ACCESS-CM2"]`
		`# get a list of unique experiments`
		`experiments = sorted(...`

2. Please explain `histSST-1950HC` as described within ESGF.	Unfortunately I was unable to verify the histSST-1950HC dataset mentioned within the Earth System Grid Federation (ESGF) before the current response. The dataset description and further information would have given you better insight, but with what I could find: histSST-1950HC is a dataset provided within the ESGF.	Sure! Here is how the experiment `histSST-1950HC` is described within ESGF: historical SSTs and historical forcing, but with 1950 halocarbon concentrations.

3. Retrieve all models under `ssp245`, `ssp585` scenarios for variable `cLeaf` with the help of *intake-esgf*.	To retrieve the models under ssp245 and ssp585 scenarios for variable `cLeaf` using intake-esgf, you can use the following code:
	`import xarray as xr`
	`from intake_esgf import`
	`esgf_connect`
	`esgf = esgf_connect()`
	`dataset='cmip6'`
	`variable='cLeaf'`
	`experiment='ssp245,ssp585' ...`
		Ok! Here is how we can retrieve all models under ssp245, ssp585 scenarios for variable `cLeaf` using Python and *intake-esgf*:
		`from intake_esgf import`
		`ESGFCatalog`
		`# initialize the catalog`
		`cat = ESGFCatalog()`
		`experiments = ["ssp245","ssp585"]`
		`variables = ["cLeaf"]`
		`search_results =`
		`cat.search(experiment_id=experiments,`
		`variable_id=variables) ...`

4. I'm trying to search for models that have data on variables `tos` and `evspsbl` using the *intake-esgf* library in Python. Can you help?	It seems like you are trying to work with climate data from ESGF and use the `intake-esgf` library. The variables `tos` and `evspsbl` relate to sea surface temperature and the evaporative fraction of surface energy, respectively. Here is a high-level guide using CMIP5: ...	Of course! Here is how we can search for models that have data on variables `tos`, `evspsbl` using Python and *intake-esgf*:
		`from intake_esgf import`
		`ESGFCatalog`
		`# initialize the catalog`
		`cat = ESGFCatalog()`
		`variables = ["tos","evspsbl"]`
		`search_results =`
		`cat.search(variable_id=variables)`
		`...`

5. What can you tell me about the experiment `piControl-withism`?	I am not familiar with a specific experiment called `piControl-withism`. However, I can suggest some options to explore this further: ...	Here is what I know about the experiment `piControl-withism`: it is an experiment ID representing pre-industrial control plus interactive ice sheet.

for our current dataset, future evaluations on held-out data subsets will help determine whether additional epochs improve performance as the dataset grows. At present, representative test queries for BERTScore evaluation are selected to reflect common use cases, while ensuring they are not drawn directly from the training data. Moving to a more formal split will strengthen reproducibility of our evaluations in future iterations. We also plan to quantify the added value of the RAG pipeline over basic fine-tuning using relevant metrics and to explore how RAG could help adapt to future changes in APIs, such as updates to intake-esgf. Finally, we note that an interesting potential extension would be to support users providing incorrect or incomplete code as input, where the model could flag likely errors and suggest relevant fixes; this capability has not yet been tested and would require dedicated evaluation and data collection. The current stage of development has only minimally captured the perspectives of real-world inexperienced users, an integral portion of the intended audience. Thus, in future iterations we intend to implement larger-scale public testing to collect and gauge user feedback, as the assistant becomes more accessible.

By combining natural language processing with domain-specific expertise, the LLM assistant reduces technical barriers and accelerates user engagement with complex Earth system data. As the system continues to evolve, it offers a scalable, intelligent interface with the potential to become an indispensable tool for researchers, educators, and policymakers seeking to navigate ESGF's vast data holdings more efficiently and intuitively. This work highlights not only the feasibility but also the value of domain-specialized LLMs in enhancing scientific data access across disciplines.

ACKNOWLEDGMENT

This research was supported by the Earth System Grid Federation 2 - US (ESGF2-US) project, which is sponsored by the Data Management Program of the Earth and Environmental Systems Sciences Division (EESSD) of the Office of Biological and Environmental Research (BER) in the US Department of Energy (DOE) Office of Science. This manuscript has been authored at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC, under contract DE- AC05-00OR22725 with the US DOE. This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. The US government retains and the publisher, by accepting the article for publication, acknowledges that the US government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for US government purposes. DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

OPEN RESEARCH

The resources associated with this work are publicly available:

- **Code for data preprocessing, model training, and evaluation:** https://github.com/dsaedinia/esgf-llm
- **Trained models:** https://huggingface.co/dsaedi
- **Curated dataset:** https://huggingface.co/datasets/dsaedi/esgf-dataset-v0.26

REFERENCES

[1] A. Gettelman, A. J. Geer, R. M. Forbes, G. R. Carmichael, G. Feingold, D. J. Posselt, G. L. Stephens, S. C. van den Heever, A. C. Varble, and P. Zuidema, "The future of earth system prediction: Advances in model-data fusion," *Science Advances*, vol. 8, no. 14, p. eabn3488, 2022. [Online]. Available: https://www.science.org/doi/abs/10.1126/sciadv.abn3488

[2] E. C. Massoud, F. Hoffman, Z. Shi, J. Tang, E. Alhajjar, M. Barnes, R. K. Braghiere, Z. Cardon, N. Collier, O. Crompton, P. J. Dennedy-Frank, S. Gautam, M. A. Gonzalez-Meler, J. K. Green, C. Koven, P. Levine, N. MacBean, J. Mao, R. T. Mills, U. Mishra, M. Mudunuru, A. A. Renchon, S. Scott, E. R. Siirila-Woodburn, M. Sprenger, C. Tague, Y. Wang, C. Xu, and C. Zarakas, "Perspectives on artificial intelligence for predictions in ecohydrology," *Artificial Intelligence for the Earth Systems*, vol. 2, no. 4, p. e230005, 2023. [Online]. Available: https://journals.ametsoc.org/view/journals/aies/2/4/AIES-D-23-0005.1.xml

[3] D. N. Williams, G. Bell, L. Cinquini, P. Fox, J. Harney, and R. Goldstone, *Earth System Grid Federation: Federated and Integrated Climate Data from Multiple Sources*. Springer Berlin Heidelberg, 2013, p. 61–77. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-37244-5_7

[4] L. Cinquini, D. Crichton, C. Mattmann, J. Harney, G. Shipman, F. Wang, R. Ananthakrishnan, N. Miller, S. Denvil, M. Morgan, Z. Pobre, G. M. Bell, C. Doutriaux, R. Drach, D. Williams, P. Kershaw, S. Pascoe, E. Gonzalez, S. Fiore, and R. Schweitzer, "The earth system grid federation: An open infrastructure for access to distributed geospatial data," *Future Generation Computer Systems*, vol. 36, p. 400–417, Jul. 2014. [Online]. Available: http://dx.doi.org/10.1016/j.future.2013.07.002

[5] Earth System Grid Federation (ESGF), "Esgf metagrid," https://esgf-node.ornl.gov/search, 2025, accessed: 2025-08-28.

[6] "intake-esgf documentation," https://intake-esgf.readthedocs.io/en/latest/, 2023, accessed: 2025-08-28.

[7] K. E. Taylor, M. Juckes, V. Balaji, L. Cinquini, S. Denvil, P. J. Durack, M. Elkington, E. Guilyardi, S. Kharin, M. Lautenschlager, B. Lawrence, D. Nadeau, and M. Stockhause, "Cmip6 model output metadata requirements, data reference syntax (drs) and controlled vocabularies (cvs)," Sep. 2018. [Online]. Available: https://doi.org/10.5281/zenodo.12768887

[8] OpenAI, J. Achiam, S. Adler *et al.*, "Gpt-4 technical report," 2023. [Online]. Available: https://arxiv.org/abs/2303.08774

[9] S. Pal, M. Bhattacharya, S.-S. Lee, and C. Chakraborty, "A domain-specific next-generation large language model (llm) or chatgpt is required for biomedical engineering and research," *Annals of Biomedical Engineering*, vol. 52, no. 3, p. 451–454, Jul. 2023. [Online]. Available: http://dx.doi.org/10.1007/s10439-023-03306-x

[10] M. Warner, B. Sharma, U. Bhatia, and A. Ganguly, *Evaluation of Cascading Infrastructure Failures and Optimal Recovery from a Network Science Perspective*. Springer International Publishing, 2019, p. 63–79. [Online]. Available: http://dx.doi.org/10.1007/978-3-030-14683-2_3

[11] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey *et al.*, "The llama 3 herd of models," 2024. [Online]. Available: https://arxiv.org/abs/2407.21783

[12] P. J. Durack, K. E. Taylor, P. J. Gleckler, G. A. Meehl, B. N. Lawrence, C. Covey, R. J. Stouffer, G. Levavasseur, A. Ben-Nasser, S. Denvil, M. Stockhause, J. M. Gregory, M. Juckes, S. K. Ames, F. Antonio, D. C. Bader, J. P. Dunne, D. Ellis, V. Eyring, S. L. Fiore, S. Joussaume, P. Kershaw, J.-F. Lamarque, M. Lautenschlager, J. Lee, C. F. Mauzey, M. Mizielinski, P. Nassisi, A. Nuzzo, E. O'Rourke, J. Painter, G. L. Potter, S. Rodriguez, and D. N. Williams, "The coupled model intercomparison project (cmip): Reviewing project history, evolution, infrastructure and implementation," *EGUsphere*, vol. 2025, pp. 1–74, 2025. [Online]. Available: https://egusphere.copernicus.org/preprints/2025/egusphere-2024-3729/

[13] E. C. Massoud *et al.*, "Hosting downscaled climate data products in esgf2-us: Expectations, requirements, and community standards," Abstract GC23E-0310, American Geophysical Union Fall Meeting, 2024.

[14] V. B. Parthasarathy, A. Zafar, A. Khan, and A. Shahid, "The ultimate guide to fine-tuning llms from basics to breakthroughs: An exhaustive review of technologies, research, best practices, applied research challenges and opportunities," 2024. [Online]. Available: https://arxiv.org/abs/2408.13296

[15] F. Trad and A. Chehab, "Prompt engineering or fine-tuning? a case study on phishing detection with large language models," *Machine Learning and Knowledge Extraction*, vol. 6, no. 1, p. 367–384, Feb. 2024. [Online]. Available: http://dx.doi.org/10.3390/make6010018

[16] Z. Han, C. Gao, J. Liu, J. Zhang, and S. Q. Zhang, "Parameter-efficient fine-tuning for large models: A comprehensive survey," 2024. [Online]. Available: https://arxiv.org/abs/2403.14608

[17] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," 2021. [Online]. Available: https://arxiv.org/abs/2106.09685

[18] M. H. Daniel Han and U. team, "Unsloth," 2023. [Online]. Available: http://github.com/unslothai/unsloth

[19] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, 2020. [Online]. Available: http://dx.doi.org/10.18653/v1/2020.emnlp-demos.6

[20] K. Singhapoo, A. Inthanil, and A. Pillai, "Fine-tuning ai models with limited resources," in *2025 11th International Conference on Engineering, Applied Sciences, and Technology (ICEAST)*. IEEE, May 2025, p. 148–151. [Online]. Available: http://dx.doi.org/10.1109/ICEAST64767.2025.11088202

[21] Unsloth, "Fine-tuning llms guide — unsloth documentation," https://docs.unsloth.ai/get-started/fine-tuning-llms-guide, n.d., accessed: 2025-08-30.

[22] Meta, "Model cards and prompt formats – llama 3.1," https://www.llama.com/docs/model-cards-and-prompt-formats/llama3_1/, n.d., accessed: 2025-08-28.

[23] Hugging Face, "Hugging face: The ai community building the future," https://huggingface.co, accessed: 2025-08-30.

[24] G. Gerganov, "llama.cpp: Port of facebook's llama model in c/c++," https://github.com/ggerganov/llama.cpp, 2023, introduced the GPT-Generated Unified Format (GGUF).

[25] F. S. Marcondes, A. Gala, R. Magalhães, F. P. de Britto, D. Durães, and P. Novais, "Using ollama," in *Natural Language Analytics with Generative Large-Language Models: A Practical Approach with Ollama and Open-Source LLMs*. Cham: Springer, 2025, pp. 23–35.

[26] T. J. Baek, "Open webui," GitHub repository and official website, 2025, extensible, self-hosted web UI for offline LLM interaction. [Online]. Available: https://github.com/open-webui/open-webui

[27] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi *et al.*, "Retrieval-augmented generation for large language models: A survey," *arXiv preprint arXiv:2312.10997*, vol. 2, no. 1, 2023.

[28] P. Béchard and O. M. Ayala, "Reducing hallucination in structured outputs via retrieval-augmented generation," *arXiv preprint arXiv:2404.08189*, 2024.

[29] K. Juvekar and A. Purwar, "Introducing a new hyper-parameter for rag: Context window utilization," *arXiv preprint arXiv:2407.19794*, 2024.

[30] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. [Online]. Available: https://arxiv.org/abs/1908.10084

[31] C. Yin and Z. Zhang, "A study of sentence similarity based on the all-minilm-l6-v2 model with "same semantics, different structure" after fine tuning," in *Proceedings of the 2024 2nd International Conference on Image, Algorithms and Artificial Intelligence (ICIAAI)*, Oct 2024, pp. 677–684.

[32] K. Rangan and Y. Yin, "A fine-tuning enhanced rag system with quantized influence measure as ai judge," *Scientific Reports*, vol. 14, no. 1, p. 27446, 2024.

[33] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with bert," *arXiv preprint arXiv:1904.09675*, 2019.

[34] N. Cooper and T. Scholak, "Perplexed: Understanding when large language models are confused," *arXiv preprint arXiv:2404.06634*, 2024.