Adaptive chatbots for Remote Sensing Education: A natural language query.

EGU General Assembly 2025 - Session EOS2.1

Rodrigo Pascual, José Francisco Díez-Pastor, Pedro Latorre-Carmona, José Manuel Aroca-Fernández

University of Burgos
Escuela Politécnica Superior
Ingeniería Informática, Spain

April 10, 2025

Overview

The Challenge in Geospatial Education

- **Complexity Hook:** Geospatial data, particularly formats like NetCDF in Earth Sciences, present significant complexity.
- **Problem:** Steep learning curve for students and new researchers.
- **Traditional Barriers:**
  - Need for programming skills (Python, R).
  - Understanding complex data structures (dimensions, variables, groups, metadata).
  - Significant time investment for basic analysis.
- **Session Relevance (EOS2.1):** Highlights the need for innovative tools to improve teaching and learning in Higher Education for Earth Sciences.

Our Approach: An LLM-Powered Assistant

The Solution

We present an LLM-powered chatbot designed specifically for remote sensing education and NetCDF analysis.

- **Core Goal:** To simplify an efficient interaction with complex NetCDF data via natural language, enhancing accessibility, efficiency, and privacy.

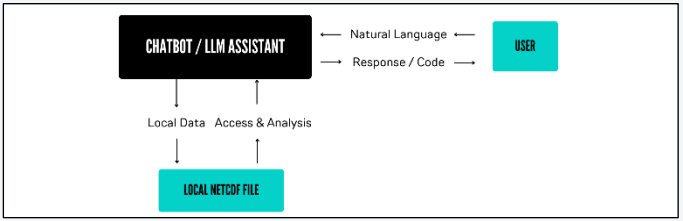


Diagram: User ↔ *app*

System Overview

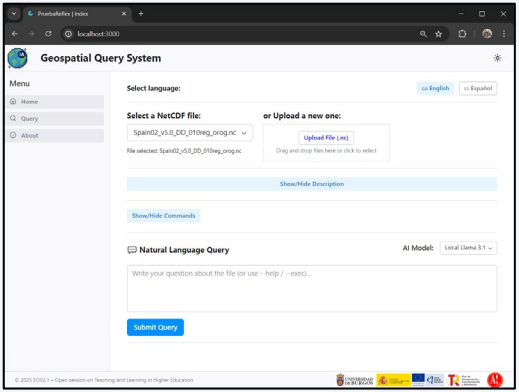
Workflow

1. User selects/uploads their local NetCDF file.
2. User selects preferred AI model (Cloud/Local).
3. User asks questions or gives commands in natural language.
4. System processes query using file context & chosen LLM.

Key Message

Intuitive interaction without initial coding required.

Application UI



Demystifying NetCDF Complexity

The Challenge

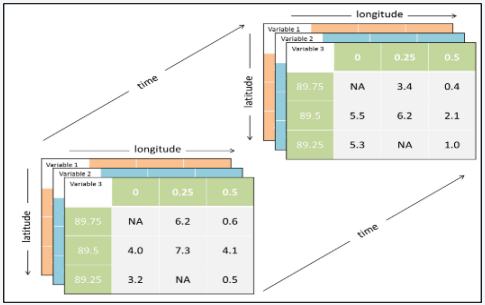


Figure: Complex NetCDF Structure

Our Tool's Solution

Hello Rodrigo,

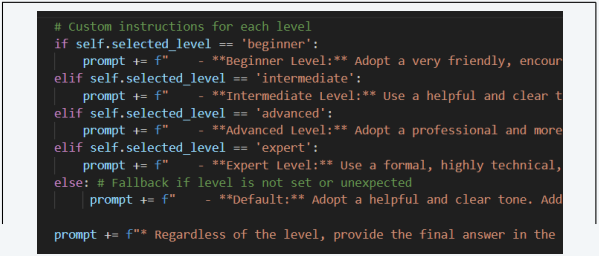
Let's break down the structure of this NetCDF file (`ndvi_output_2023.nc`). Think of it like a well-organized data container, designed to store gridded data that changes over time. Here's a simplified outline of its structure:

```
NetCDF File (ndvi_output_2023.nc)
├── Global Attributes:  (Metadata describing the entire file)
│   ├── Conventions: CF-1.9  (Standard used)
│   ├── institution: openEO platform... (who created it)
│   ├── description:  (Empty in this case)
│   └── title: (Empty in this case)
│
├── Dimensions: (Sizes of the axes/grids)
│   ├── t: 108 (Time dimension - 108 time steps)
│   ├── x: 2073 (X dimension - usually longitude or a projected X coordinate)
│   └── y: 1682 (Y dimension - usually latitude or a projected Y coordinate)
│
└── Variables: (The actual data and coordinate information)
    ├── t: (Time values)
    │   ├── Attributes: units (days since 1990-01-01), etc.
    │   └── Data: [12053, 12055, ...] (representing dates)
    ├── x: (X coordinate values)
    │   ├── Attributes: units (m), etc.
```

Figure: Automated File Summary

Adaptive Learning &Personalization

- **Concept:** User profile setup (Name, Geoscience Level).
- **Mechanism:** Dynamic prompt adjustment based on level:
  - *Beginner/Intermediate:* Friendlier tone, simpler explanations, optional name usage.
  - *Advanced/Expert:* More technical vocabulary, professional tone.
- **Educational Benefit:** Tailors the learning experience, making it less intimidating for beginners and more efficient for experts.



```
# Custom instructions for each level
if self.selected_level == 'beginner':
    prompt += f"    - **Beginner Level:** Adopt a very friendly, encour
elif self.selected_level == 'intermediate':
    prompt += f"    - **Intermediate Level:** Use a helpful and clear t
elif self.selected_level == 'advanced':
    prompt += f"    - **Advanced Level:** Adopt a professional and more
elif self.selected_level == 'expert':
    prompt += f"    - **Expert Level:** Use a formal, highly technical,
else: # Fallback if level is not set or unexpected
    prompt += f"    - **Default:** Adopt a helpful and clear tone. Add

prompt += f"* Regardless of the level, provide the final answer in the
```

Transparency, Skill Building & Privacy

Code Generation ('--exec')

- Generated Python code is shown to the user.
- **Benefit 1 (Transparency):** See *how* the task is performed.
- **Benefit 2 (Learning):** Learn Python/library syntax.
- **Benefit 3 (Control):** Copy, modify, execute code.

Local Models Tested

- llama 3.1 & deepseek-r1:7b

Data Privacy & Local LLMs

Crucial Point

System interacts directly with user's files.
User datasets remain entirely local.

Examples

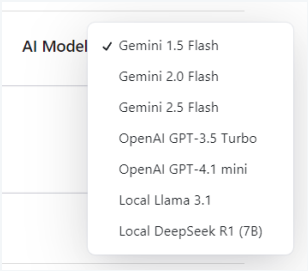
Benefits of Local Models

- Reinforces privacy.
- Enables offline use.
- Reduces cloud API reliance/cost.

Model Performance Comparison (Preliminary)

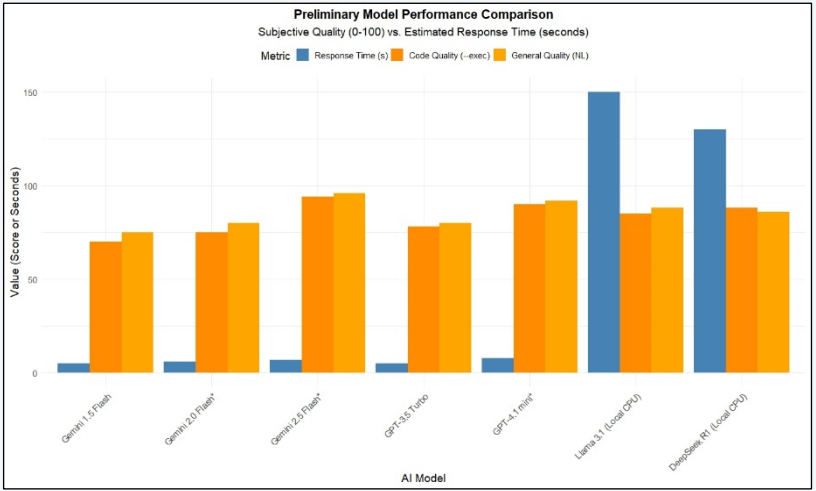
Qualitative Assessment: Response Quality vs. Time

Preliminary tests comparing available models highlight the following trade-offs, especially considering the local execution environment (16GB RAM, CPU, no dedicated CUDA):



Model Type	Key Advantages	Key Disadvantages / Considerations
Cloud (Gemini, OpenAI)	• Generally higher response quality (esp. complex tasks). • Access to latest model versions. • Fast cloud computation	• API dependency/costs. • Data sent to cloud provider (privacy concern).
Local (Llama 3.1, DeepSeek R1)	• Enhanced data privacy (data stays local). • Offline capability \& no direct API costs.	• Significantly higher response time on CPU (>100s). • Requires sufficient local RAM/Hardware.

Model Performance Comparison (Preliminary)



Video Demonstration (8 min)

(Live narration during video playback)

Placeholder for Video Screenshot

Key Takeaways & Impact

Main Benefits

- Lowers barrier to entry for complex NetCDF data.
- Provides adaptive, personalized learning.
- Enhances understanding (NL + Code Transparency).
- Prioritizes data privacy (Local files + LLMs).

Architecture

- Built using Python web framework: **Reflex**.
- RAG-inspired context injection for LLMs.

Potential Impact

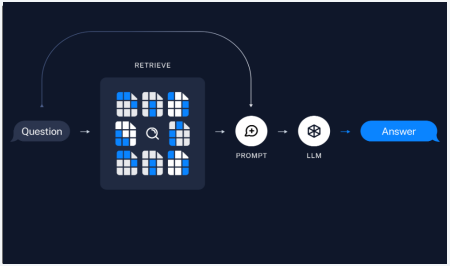


Figure: https://python.langchain.com/docs/tutorials/rag/

Example: RAG-like Workflow

Future Directions

- **Enhance RAG:** Deeper context integration (e.g., data chunking for very large files).
- **Direct Visualization:** Generate plots directly within the UI.
- **Evaluation:** Conduct formal user studies to quantify learning impact.
- **Model Support:** Expand compatibility with more models and explore fine-tuning possibilities.
- **Error Handling:** More granular feedback on code execution errors.

Thank you for your attention

Rodrigo Pascual, José Francisco Díez-Pastor, Pedro Latorre-Carmona, José Manuel Aroca-Fernández

University of Burgos
Escuela Politécnica Superior
Ingeniería Informática, Spain

April 10, 2025