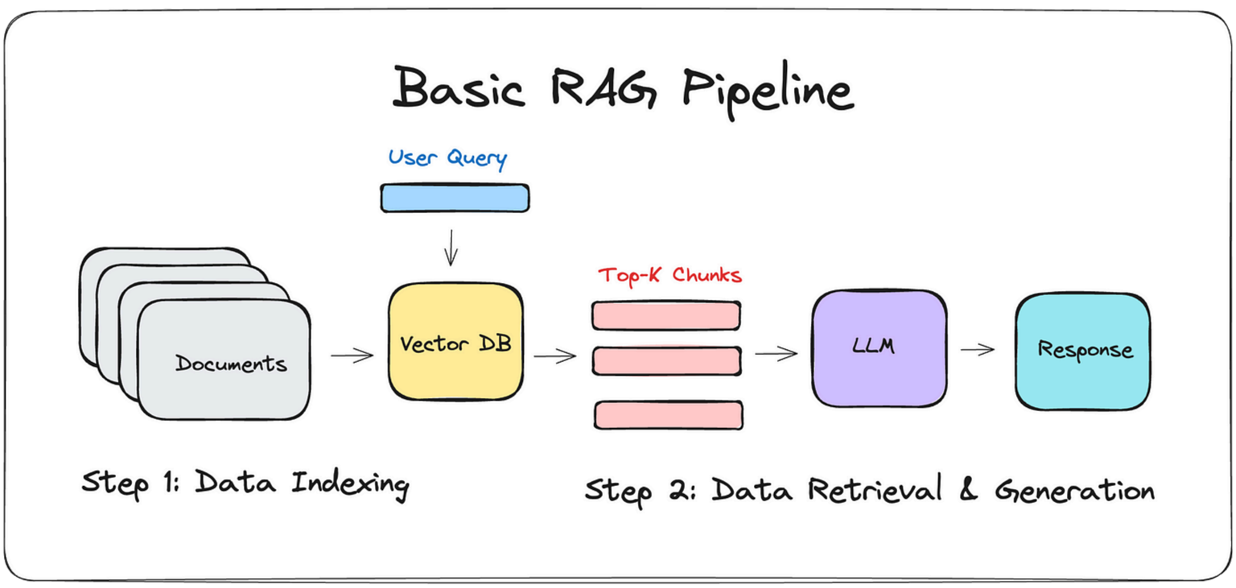
Code The Change x The Nature Conservancy
RAG for Agroforestry Final Report

May - July 2025

Introduction to RAG

Retrieval-Augmented Generation (RAG) is a method that enhances the capabilities of Large Language Models (LLMs) by grounding their responses in external knowledge. This allows the model to answer highly specific domain-based questions more accurately and reliably. In our project, we focused on building a RAG system designed specifically for agroforestry, using research papers and annotated datasets curated by scientists at The Nature Conservancy (TNC). By combining traditional information retrieval with LLM-driven generation, the RAG system aims to answer questions related to soil health, carbon content, land types, and more contextualized within real, scientific data.



Overview of the Project

Our goals for this project largely focus on these two areas:

- Develop a robust RAG system trained on annotated agroforestry articles provided by TNC scientists.
- Evaluate and benchmark the system's performance in terms of accuracy and confidence.

Model Specifics

We started by looking at the user's query. To help the system understand what the question is about, we used a tool called spaCy to pull out important details like the location mentioned, the type of planting, or any specific organizations. These details help narrow down what kind of answer the person is looking for, and eliminate articles which don't mention this information.

Next, we needed to find the most relevant documents from our collection of agroforestry research papers. This is what's called the retrieval step in RAG. Instead of just matching exact words, we turned both the question and the documents into numerical representations, making it easier to measure how closely related they are semantically. We employed a tool called Hugging Face's sentence-transformers for this.

To actually retrieve the relevant articles, we created a scoring system. First, we filtered out any articles that didn't mention the most important details from the question. Then we gave scores to the remaining articles based on how many relevant words they shared with the question and how closely related their meaning was. The top five articles with the highest scores were selected to move forward.
Once the relevant articles were identified, we segmented them into smaller units—referred to as chunks—to ensure they could be efficiently processed by the language model, which has a limited input capacity. Each chunk was then re-evaluated against the original question using semantic similarity measures, and the most relevant segments were selected for response generation. However, this approach presented challenges: in some cases, critical information was distributed across multiple chunks or embedded within tables that exceeded the model's token limits. As a result, the system occasionally failed to retrieve the necessary context to generate an accurate response.

Finally, in the generation step, we passed both the selected chunks and the original question into a language model to generate an answer. We used examples (called "few-shot prompting") to guide the model in how to respond. We also tested different language models to see which one worked best factoring in how accurate they were, how much text they could handle at once, and whether we could run them locally or not.

Evaluating the Model

To benchmark the system's performance, we created a Python script that automatically generated structured questions based on a master spreadsheet ("S3 Measurements"). For instance, one question might be: "What is the total soil organic carbon of planting woodlot in Central Kenya?" Each of these ~700 questions was passed through the RAG pipeline, and the system's answer was compared to the ground truth using semantic similarity metrics.

This benchmarking process revealed both strengths and weaknesses. On one hand, the model performed reasonably well in matching high-level semantics. On the other hand, it struggled with nuanced scientific values, especially when the answer was expressed in different units. Additionally, the runtime for processing hundreds of queries was quite long, averaging 5–6 hours. These factors introduced variability in our reported 50% accuracy and suggested the need for better evaluation techniques that account for unit mismatches and structured data like tables.

Key Findings

Below is an example output of one of our benchmarking questions. Our question is composed by taking the entries in the ‘variable.name’, ‘refor.type’, and 'site.sitename' entries in each row from the human inputted agroforestry spreadsheet. The format of the question is: "What is the {‘variable.name’} of planting {‘refor.type’} in {'site.sitename'}?" By excluding rows with invalid or empty entries, we were able to create a benchmarking dataset with around 1700 unique questions, each having its unique answer.

We recorded the “true_answer”, which comes from combining the entries in ‘mean.in.original.units’ and ‘original.units’. Then we used a semantic similarity function to compare the model’s answer with the true answer.

```python
{
    "question": "What is the aboveground_carbon of planting Intensive shade coffee in La Antigua river?",
    "true_answer": "30.62201 MgC/ha",
    "model_answer": "25.9 Mg/ha (Gilroy et al., 2014)",
    "similarity": 0.9394
  }
```

After running our benchmarking dataset on a pre-trained Llama 3 model with 8 billion parameters and using the Multi Qa Mpnet Base Dot V1 model as our embedding model and the All MiniLM L6 V2 as our embedding function for creating the vector database, we recorded the accuracy of our RAG-LLM pipeline to be around 50%. However, this number may change depending on the LLM and embedding models and functions used.

While parsing through the thousands of PDFs of research papers, we found that people structured their papers in many different ways. This variety made it more difficult to retrieve the most relevant parts of each article, since we don’t exactly know where each section is located. Therefore, we think that it would be helpful to propose a standardized formatting for research papers, so that they could be best read by RAG-powered LLMs. This could potentially significantly improve the retrieval accuracy.

When going through each of the steps of RAG, we noticed that some tools worked better than others. For instance, we decided to move our code to a cloud-based environment, namely Google Colab, instead of using a local code editor like Visual Studio Code (VSCode). We found that this helped alleviate pressure from our local computing resources (like the hardware in our laptops), and gave us better access to using a more powerful paid cloud computing environment. The Colab environment is also easier to use, requiring a less steep learning curve than running code on local editors. This would help our LLM be more accessible to researchers who may not have a technical background.

Various different LLMs have a range of different responses to the prompt; The first filter we apply when choosing which LLM to use is the context window.

A brief comparison of each LLM is shown in the table below:

Model Name	Source	Local Deployment	Context Window	API Cost (if applicable)	Notes / How to Run
GPT-4o	Proprietary	No	~128K tokens	$5–$30 per million tokens	Best-in-class; use via OpenAI API
Claude 3.5 Sonnet	Proprietary	No	~200K tokens	Tiered (Anthropic API)	Strong RAG grounding; Anthropic API only
Gemini 1.5 Flash	Proprietary	No	~1M tokens	Free/Paid (Google Cloud)	Fast, efficient; API-only deployment
xAI Grok-2	Proprietary	No	~128K tokens	Subscription-based	Via X platform; strong multi-turn reasoning
LLaMA 3.1 (70B)	Open-source	Yes	128K tokens	Free (local), API optional	Run via Ollama, llama.cpp, or HuggingFace
Mistral Large 2	Open-source	Yes	128K tokens	Free (local), API optional	Efficient, strong open-source option
DeepSeek-R1	Open-source	Yes	~128K tokens	Free	Great for reasoning; run locally via HuggingFace
Qwen 3	Open-source	Yes	~128K tokens	Free	Strong multilingual support; HF/Ollama compatible
DeepSeek-MoE	Open-source	Yes	~128K tokens	Free	MoE architecture; performant in retrieval settings
Claude 3.5 Opus	Proprietary	No	~200K+ tokens	Premium pricing	Stronger than Sonnet; ideal for benchmark upper bound

Some example responses from models which piqued particular interest are shown below:

1. **Model: LLama 3.1 (8b)**
   User Question: What is the percentage soil organic carbon of planting Wheat & R. Prinode intercropping in Ganta-Afoshum?
   Answer:
   11.8% (cleaned_Gebremeskel 2018)
   Notes: This is an ideal response for this intended task; It provides a cited source as well as the correct answer from the relevant research article.
2. **Model: Deepseek-r1 1.5B**
   User Question: What is the understory_carbon of planting silvopasture in Villaflores?
   Answer:
   Okay, so I need to figure out the understory carbon of planting silvopasture in Villaflores. Looking at the provided excerpts, I see a lot of tables about carbon storage and tree diversity in areas called Villaflores and Suchiapa.

   First, I should look for any tables that mention silvopasture. Scanning through, I see a table about carbon storage in different systems, but that's more about the total carbon stored rather than understory carbon.

Then there's a table comparing tree AGB and grass AGB between silvopasture and open pasture systems in both Villaflores and Suchiapa. Silvopasture seems to be the system in question here.

In the table, under Silvopasture, I see the following values:
- Tree AGB: 10.75 (1.79) a
- Grass AGB: 0.78 (0.15) bA
- Tree RB: 2.53 (0.41) a
- Grass RB: 1.47 (0.29) aA
- Litter (Oi horizon): 0.546 (0.11) a
- Litter (Oe horizon): 0.617 (0.12) a
- Litter (Oa horizon): 0.938 (0.04) a
- SOC (0-15 cm): 89.72 (5.54) aB
- Total C stock: 104.82

Looking at the AGB values, Silvopasture has 10.75 Mg C ha-1. This seems to be the understory carbon for the silvopasture system in Villaflores.

I don't see any other silvopastural system data in the provided tables, so I can be confident that this is the value they're asking for. Therefore, the understory carbon is 10.75 Mg C ha-1.
</think>

The understory carbon of planting silvopasture in Villaflores is 10.75 Mg C ha-1.

Notes: This response is very typical with a lot of the LLMs for this intended task; It does end up with a final answer which is within 3-5 Mg C ha-1 of the correct value but it does so with a lot more reasoning and output tokens used up as a result. The accuracy of these longer answers also differs from model to model.

Next Steps and Key questions

Because RAG is a modular process, we can easily swap out components at each stage without disrupting the overall flow. With more time, this flexibility would allow us to test various open and closed-source LLMs, vector databases, and sentence transformers to identify the best combination for handling data-intensive papers.

As seen from our benchmarking, there is still room to improve when answering questions that require specific numerical outputs. We believe that this can be resolved by further improving our retrieval stage, specifically when selecting the most relevant chunks from retrieved articles. One question we asked is, “Where do the most important pieces of numerical data often appear in these research papers?” After flipping through a few papers, we noticed that they often appeared on tables or graphs within the papers. However, we wonder if there are other potential spots that may be important to focus on. We believe that by improving the retrieval stage, we could boost the model's overall performance.

Based off our observations from our retrieval stage and overall codebase, we propose the following directions for further work:

- Improving Information Retrieval: Exploring more advanced indexing and retrieval techniques (e.g., using vector databases) to handle larger knowledge bases and improve retrieval accuracy.
- Enhancing Text Processing: Investigating more sophisticated chunking and reranking strategies to ensure the most relevant information is extracted and presented to the LLM.
- Optimizing LLM Interaction: Experimenting with different prompting techniques, LLM models, and parameters to improve the quality and conciseness of the generated responses.
- Handling Diverse Data Formats: Extending the system to handle information from various data formats beyond JSON, such as PDFs, spreadsheets, or databases.
- User Interface: Developing a more user-friendly interface for interacting with the RAG system.

Challenges and Issues

One of the main challenges we ran into was input limits on the LLMs we were using. For context, each LLM has its own input limit (or better known as token limit) that prevents users from passing in too long of a prompt at one time. For example the context window of the Llama3 was smaller than the length of a lot of the articles we were retrieving, and while using it we often got errors that told us the model could not be run because our prompt was too long. For us, we decided to implement our chunking technique of further breaking down articles, as well as using models with larger token limits.

Another issue we ran into was choosing the right LLM for the job. We noticed that some models tended to be more wordy, like the Deepseek R1 model. Instead of outputting a specific numerical answer, even when prompted to do so, the R1 often liked to describe its thought process in a long paragraph, and only gave out an answer at the end of the response. Some other models we tested, such as the Llama3 model, didn't struggle with this issue.

Additionally, it seemed difficult for LLMs to process such large amounts of text. Often, when the relevant context was present in the information passed into the LLM it still wasn't able to parse through and choose the correct numbers from the tables and text provided.

Instructions

The entirety of our code is in the github repository linked here:

Prerequisites:

Before you execute any code in the notebook, there are two things to arrange: a Hugging Face access token and the project's data folder. The notebook assumes both are already in place.

Hugging Face Access Token

1) Head over to https://huggingface.co, sign in (or create a free account), and open Settings → Access Tokens. Click New token, pick the Read role, and let the site generate a string that starts with hf_.
2) Copy it: this single line tells the Hugging Face API that you're allowed to download models and embeddings.
3) In Google Cola, open the left sidebar, choose ⁝ More → Settings → Secrets, and add a new entry named **HUGGINGFACE_HUB_TOKEN** with the value you just copied.

The "RAG Materials" Folder

1) All source articles and the benchmark JSON reside in a directory called RAG Materials. The retrieval code expects to find that folder intact, with sub-files untouched.
2) If you are on Colab, simply drag the entire folder into the root of your Google Drive. When the notebook runs its first cell to mount Drive (drive.mount('/content/drive')), the path /content/drive/MyDrive/RAG Materials will appear. **Note**: When the colab is running you will likely see a popup asking you for permission to access files in your drive.

Conclusion

Overall, our Google Colab notebook provides a solid foundation for building a RAG system for domain-specific question answering, highlighting the power of combining retrieval and generation for improved accuracy and trustworthiness.

RAG-powered LLMs amplify the strengths of LLMs by providing the model additional context to provide domain specific information quickly to researchers. We believe that this technology could improve the research process by expediting the literature review while ensuring access to up-to-date information. Moreover, the versatility of RAG makes it applicable across various academic fields, not just agroforestry.

Acknowledgements

We would like to express our gratitude to Professor Charlotte Chang of Pomona College for guiding us throughout this project, as well as the Spring 2024 Code the Change team for creating the baseline for which we created off of. We would also like to thank Dr. Susan Cook-Patton and her colleagues at TNC for compiling and labeling the critical articles that we based our retrieval from. We are also deeply appreciative of Dr. Cook-Patton's continued support in providing expert guidance on the scientific aspects of this project. We're also grateful for Pomona College's financial support for the entire Code the Change X TNC summer team. Last but not least, we would also like to recognize our team lead Laura Vairus and fellow team members Wendi Zheng and Alyssa Wu for their invaluable support and collaboration during group meetings.

Citations

In the process of preparing to build the RAG pipeline, Prof Chang compiled a list of previous research on both agroforestry and natural language processing to help us better understand industry needs and find optimal procedures. Below is a list of papers we found especially helpful.

Bukoski, Jacob J., Susan C. Cook-Patton, Cyril Melikov, Hongyi Ban, Jessica L. Chen, Elizabeth D. Goldman, Nancy L. Harris, and Matthew D. Potts. 2022. "Rates and Drivers of Aboveground Carbon Accumulation in Global Monoculture Plantation Forests." *Nature Communications* 13 (1). https://doi.org/10.1038/s41467-022-31380-7.

Cook-Patton, Susan, Tanushree Biswas, Rémi Cardinael, Katherine Culbertson, Andrea DeStefano, Edenise Garcia, Michael Jacobson, et al. 2022. "Building a Global Carbon Database to Characterize Agroforestry as a Natural Climate Solution - Agritrop." *Cirad.fr*. http://agritrop.cirad.fr/601881/.

Gao, Yunfan, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang and Haofen Wang. "Retrieval-Augmented Generation for Large Language Models: A Survey." ArXiv abs/2312.10997 (2023): n. pag.

Hart, Darren M, Samantha Yeo, Maya Almaraz, Damien Beillouin, Rémi Cardinael, Édenise Garcia, Sonja Kay, et al. 2023. "Priority Science Can Accelerate Agroforestry as a Natural Climate Solution." *Nature Climate Change* 13 (11): 1179–90. https://doi.org/10.1038/s41558-023-01810-5.

Lehman, Eric P., Evan Hernandez, Diwakar Mahajan, Jonas Wulff, Micah J. Smith, Zachary M. Ziegler, Daniel Nadler, Peter Szolovits, Alistair E. W. Johnson and Emily Alsentzer. "Do We Still Need Clinical Language Models?" ArXiv abs/2302.08091 (2023): n. pag.

Lewis, Patrick, Ethan Perez, Aleksandara Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel and Douwe Kiela.

"Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." ArXiv abs/2005.11401 (2020): n. Pag.

Xiong, Guangzhi, Qiao Jin, Zhiyong Lu and Aidong Zhang. "Benchmarking Retrieval-Augmented Generation for Medicine." ArXiv abs/2402.13178 (2024): n. Pag.