CCEX Climate - Capstone Report

Author List:
Khoa Vo - hvvo1@uci.edu
Jasmine Pham - thaotp3@uci.edu
Ameya Gaitonde - ameyag@uci.edu
Xizhou Liu - xizhuol@uci.edu
Veronica Wang - xinyw27@uci.edu

Project partner information
California Climate Exchange (CCEX)
Li Qiu - li@ccex.co

Abstract

We develop and implement a retrieval-augmented generation (RAG) model to create Project Design Documents (PDDs) for carbon credit certification through the Verra registry. The model uses query decomposition, vector-based similarity search, and reranking techniques to efficiently retrieve relevant PDDs. By integrating these documents, the model ensures that project proposals meet the necessary requirements for carbon credit issuance. To qualitatively assess the model's effectiveness, we performed validation by comparing PDDs generated with and without the model. Finally, the RAG model was deployed within a chatbot interface built using Streamlit, offering an interactive and user-friendly tool for clients seeking to draft compliant PDDs. The results highlight the potential of RAG-based approaches for automating and speeding up the creation of carbon credit documentation.

Background Information

In 2023, California passed SB 253, a law that requires businesses with over $1 billion in revenue to report their carbon emissions to the state. Most companies don't have an ESG (Environmental, Social, and Governance) team, so they have to deal with the additional cost of hiring and training ESG team members to analyze the company's carbon emissions.

Under California's Cap-and-Trade program established by the California Air Resources Board, there exists a statewide "cap" on the total amount of greenhouse gases that certain industries can emit. Each company is required to hold an amount of carbon credits that is equivalent to the amount of carbon dioxide that they need to emit. One carbon credit permits its owner to emit one metric ton of greenhouse gas emissions.

Carbon credits are tradable instruments. For example, if a company emits 5 metric tons of greenhouse gas emissions less than what is permitted under the cap, they have 5 carbon credits that can be sold on the carbon credit market. Likewise, if they emit 5 metric tons *more* than their cap, they are required to purchase an equivalent number of carbon credits or face significant financial penalties.

One way companies can earn carbon credits is by implementing carbon offset projects and registering them with Verra, one of the largest carbon crediting programs in the world. These typically involve renewable energy and reforestation projects. After the project is

registered with Verra, its greenhouse gas reductions are verified and confirmed, allowing the project proponent to be issued carbon credits.

Before the project is verified, the project proponent must submit a project description document (PDD) that details the project location, project type, local partners, baseline emissions estimates, and compliance with Verra standards and local laws and regulations.

Our project partner informed us that it takes a few months to write up a preliminary PDD, and that it is essential to speed up this process given the growing carbon credits market. Our partner asked us to develop an AI agent that could shorten this timeframe from months to potentially just a few weeks.

Problem statement

Given historical Project Design Documents (PDDs) from the Verra registry, our goal is to develop an AI agent powered by a retrieval-augmented generation (RAG) model to assist users in creating accurate PDDs based on their input. Users interact with the system through a chatbot interface, answering a series of questions about their project. The AI then leverages historical PDDs and other relevant documents to generate tailored recommendations and draft PDDs for users, ensuring accuracy and alignment with Verra's standards.

This project requires collecting, processing, and preparing large amounts of data from various Verra registry resources, including methodologies, validation and verification guidelines, and other program-specific documents. To achieve this, we implemented a robust data scraping pipeline to retrieve documents from the Verra registry and process them for use in the RAG model.

- **Data Collection:** The data collection process aimed to gather comprehensive datasets from the Verra registry, including:
    a. Registry Programs: Verified Carbon Standard (VCS), Plastic Waste Reduction Program (PWRP), Climate, Community & Biodiversity Standards (CCB), Sustainable Development Verified Impact Standard (SD-VISTA), California Air Resources Board (CA_OPR)
    b. Resource Tags: Methodologies, Validation and Verification guidelines
    c. Latest Updates: News articles, program notices, Verra Views
- **Approach:** To collect the data, we designed a Python-based scraper using Selenium for dynamic web page interaction and requests for downloading files. The scraping pipeline was structured to handle the unique requirements of different Verra registry sections:
    a. Scraping Registry Programs: Documents were categorized into predefined sections, such as pipeline, registration, validation, and verification.
    b. Scraping Resource Tags: Resources such as methodologies and validation guidelines were downloaded and converted into plain text format for easy integration into the RAG pipeline.
    c. Scraping Latest Updates: News articles, program notices, and blogs under the "Latest" tag were scraped and processed. The scraper navigated through paginated content dynamically. Articles were extracted as plain text to be indexed for RAG retrieval.
    d. PDF Conversion: A dedicated module was built to convert downloaded PDF files into plain text files for compatibility with natural language processing tasks.

Relevant exploratory data analysis

The Verra registry provided a csv file with information about the location of projects, each project's status, and the methodology associated with each project. We visualized this data to get a better overview of the dataset. Figure 1 displays the top 12 countries with the most projects, where we see China and India collectively have the most projects. Figure 2 shows a breakdown of project status by region. Finally, the third figure displays the most common methodologies per region and project type.

As we built our RAG model, we needed to be as specific as possible with our queries so that the model could retrieve the most relevant documents. For example, for a reforestation project in northern Brazil, we would need the project to comply with the Brazil Forest Code, possible regional laws, and likely use the VM0007 or VM0015 methodology associated with such a project. Lastly, we focused our RAG model on registered and approved Verra documents, as this would increase the likelihood of our generated PDD gaining approval from Verra.

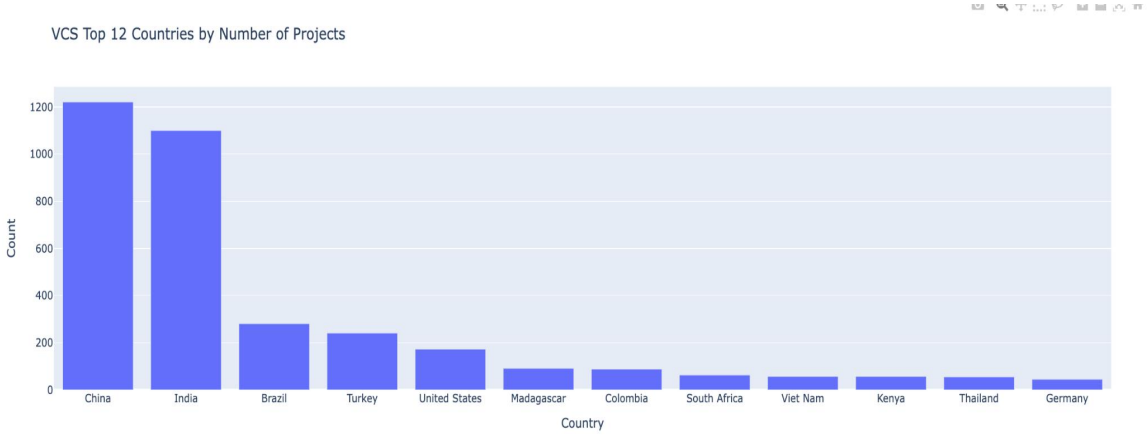


Figure 1: Top 12 countries

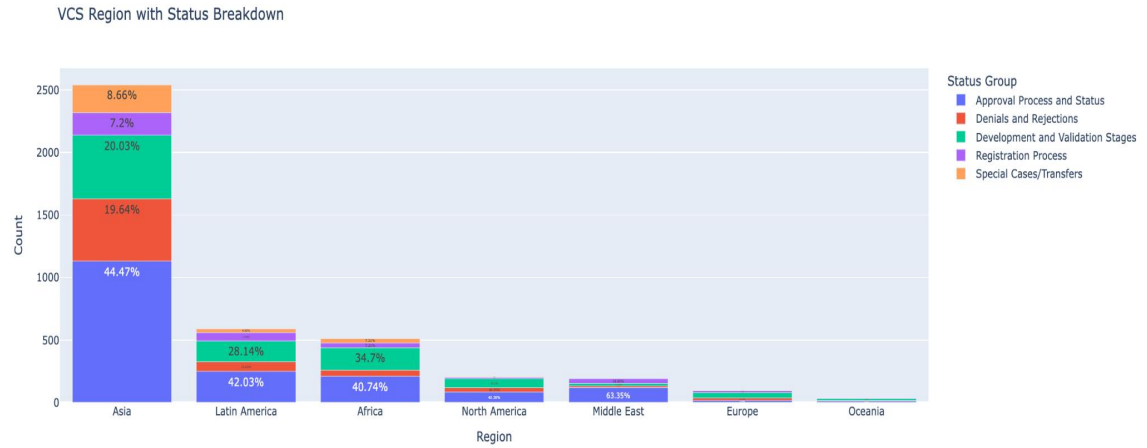


Figure 2: Region with status breakdown

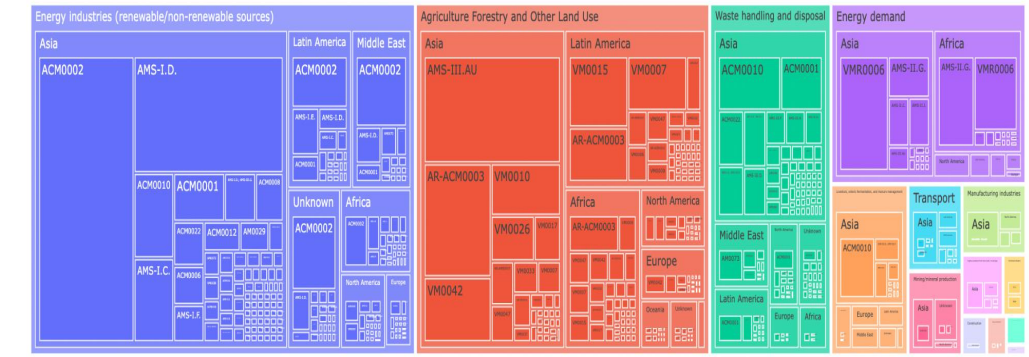


Figure 3: Common methodologies across region and project type

PDD structure:

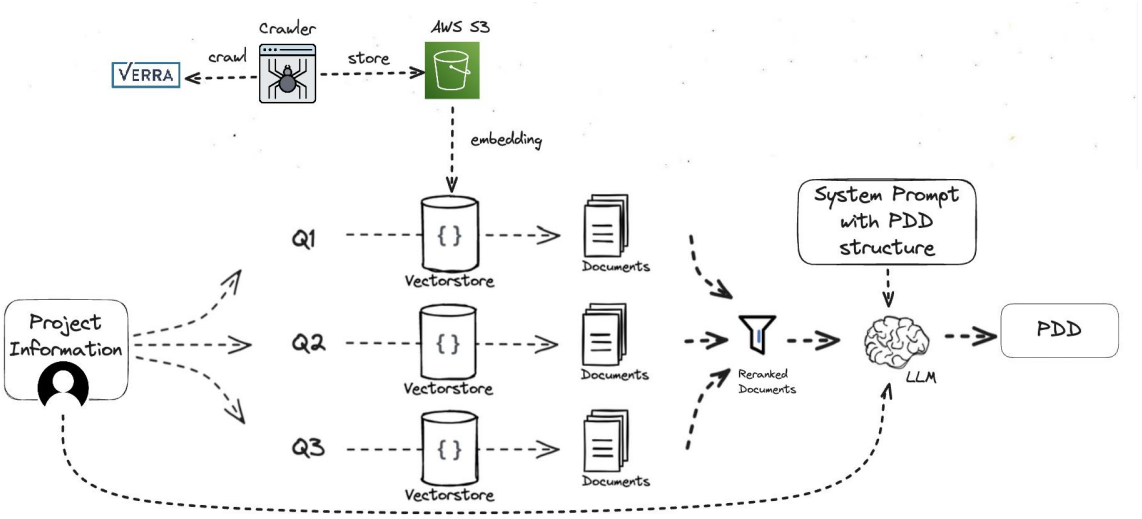
All PDDS are required to follow the same structure detailed below. While some sections of the project overview are expected to be provided to us by a client, there are other keywords like “baseline”, “net emissions”, and “leakage” that can be used to help the RAG model draw the most relevant documents and pieces of information. The PDD structure and the methodology breakdown from Figure 3 helped guide us in our construction of the RAG model, as we will describe next.

- **Project Overview:** Project type, eligibility, design, stakeholders, timeline, and estimated greenhouse gas reductions.
- **Safeguards**: Explains how the project mitigates environmental or socio-economic risks. Details compliance with local and national regulatory requirements.
- **Methodology**: Framework for emissions accounting is detailed. This includes defining the boundaries, baseline scenarios, and any deviations.
- **Emission Reductions**: Calculation of baseline, project, and net emissions, including leakage.
- **Monitoring**: Long-term tracking emissions reductions to ensure they are lasting.

Methodology

a. Overall Architecture of the project

The overall architecture of the project is illustrated in the provided diagram:



b. Crawler

The system starts with a crawler designed to extract relevant information from the Verra website. This crawler scrapes project-related data, methodologies, notices, news, blogs, etc. from the registry and stores it in a structured format. The extracted data is pushed into an AWS S3 bucket for scalable and secure storage.

c. Storage

AWS S3 serves as the central repository for all project information. Its robust capabilities for handling large datasets make it suitable for storing crawled data before further processing.

d. Embedding

Once the data is stored, the system applies embedding techniques to convert textual information into vector representations. These embeddings capture semantic meanings and serve as the foundation for efficient information retrieval. The embedded vectors are indexed into a vector store for fast and accurate query retrieval.

Embedding Model Selection

During our evaluation process, we compared several embedding models on a subset of the data (~52 PDFs, including Project Design Documents (PDDs), methodologies, and relevant news articles). This evaluation used 16 simple queries, focusing on both retrieval accuracy and computational efficiency. Among the models

tested, *BGE-large* demonstrated the highest Mean Reciprocal Rank (MRR), However, practical limitations in our cloud infrastructure limitations and embedding entire database runtime led us to select the *Snowflake*/*snowflake-arctic-embed-m-v1.5* model with chunk_size=2000 tokens instead.

Model	Chunk Size	MRR	Query Time (ms)	Model Size (MB)
BGE-large	500	0.85	26.3	2200
	1000	0.88	24.5	
	2000	0.91	22.1	
Snowflake/snowflake-arctic-embed-m-v1.5	500	0.79	10.1	490
	1000	0.81	8.9	
	2000	0.85	7.3	

e. Retrieval

The retrieval process in this project is central to the functionality of the Retrieval-Augmented Generation (RAG) framework. This step ensures that relevant documents are chosen from the vector store to provide accurate, contextually aware, and comprehensive responses to user inputs. For our project, we adopt Fusion-RAG to improve the quality of LLM-generated responses and overcome the limitation of a single query as in the traditional RAG system. The fusion RAG has the following components: query decomposition, vector-based similarity search, and reranking.

1. Query Decomposition:

Complex queries from users often have multiple facets or subtopics, which may not be adequately addressed by a single retrieval pass. Therefore, we used LLM to break down into K simpler subqueries (for our project, we use K=12). Each subqueries targets a different aspect of the project, ensuring comprehensive coverage of relevant information.

For example:

Input Query: "*How does the project using VM00015 methodology contribute to greenhouse gas reduction and what is its impact on sustainable land management?*"

Decomposed Subqueries:

Subquery 1: "*How does the project using VM00015 methodology contribute to greenhouse gas reduction?*"

Subquery 2: "*What is the impact of the project using VM00015 methodology on sustainable land management?*"

2. Vector-based similarity search:

Each subquery will be embedded into the vector through the embedding we used and interact with the vector store independently to retrieve top-K relevant documents from the vector store (K=5) .

3. Fusion and Reranking:

After retrieving relevant documents for the decomposed sub-queries, the system employs reciprocal rank fusion (RRF) to rerank the documents based on their relevance scores. Each document is assigned a reciprocal rank fusion score ($rff\ score$), which is calculated using the formula:

$$rrf\ score\ = \frac{1}{rank\ +\ k}$$

Where:

$rank$: The position of the document in the sorted list of results for a given subquery (sorted by distance or similarity).

k: A constant smoothing factor to adjust the weight assigned to lower-ranked documents, ensuring that even less prominently ranked documents contribute to the fusion process.

The reranked list is truncated to the top 30 documents based on their $rff\ score$, which are passed to the final generation phase.

f. System prompt

In the final generation phase of the Retrieval-Augmented Generation (RAG) system, the system prompt directs the large language model to generate outputs that adhere to the Project Design Document (PDD) structure identified during our exploratory data analysis (EDA). By incorporating precise instructions and a PDD-aligned template, the prompt ensures that the LLM produces responses that are well-structured, contextually relevant, and aligned with user inputs.

g. User Interface

The interface is built on Streamlit. The AI start page features a questionnaire designed to capture all the necessary information for generating a complete Project Design Document (PDD). After users input their responses, they can interact with the AI to generate the PDD. Alternatively, users can skip the questionnaire and ask specific questions about projects, the registry, or the validation and verification process. Finally, users can download the generated PDDs in PDF format, enabling easy sharing, review, and documentation.

- **Conclusion and discussion**

a. **Key findings**:

By thoroughly understanding the operations of registries, including their guidance standards, methodologies, and past projects, we identified essential elements required for generating Project Design Documents (PDDs). We observed that different types of projects often rely on specific methodologies, and these can vary based on regional requirements.

We successfully launched a demo of an AI system capable of retrieving and extracting relevant information from documents crawled from Verra. Combined with user inputs, the system generated sample PDDs that adhered to Verra's guidance and standards. These outputs can serve as a solid foundation for users to expand into complete PDDs.

b. Evaluation of methods

i. **Validation process**:

Our current validation process ensures that the Project Design Documents (PDDs) generated by our system meet the required standards and provide accurate, reliable information. The response time for generating a PDD is designed to be efficient, typically within 1-2 minutes. While this is not ideal, it still saves significant time compared to manual work. To validate the quality of these PDDs, we compare them against approved PDDs from similar sectors to ensure consistency, completeness, and adherence to industry standards.

We also establish a baseline by comparing PDDs generated by Climate AI's retrieval models with those produced by LLM that do not utilize registry-based information retrieval. Although the text generated from user input appears similar in both cases, the **experimental** PDDs generated by our system demonstrate greater relevance. This is because they integrate methodologies, standard information, and data from previously approved projects directly retrieved from verified registries. In contrast, **baseline** models often omit sections requiring specific data, such as methodologies.

To further refine our system, our future plan includes consulting with our sponsor to validate the information generated by Climate AI and conducting user testing. This will help us gather feedback on response time, accuracy, and overall satisfaction, ensuring our tool meets user needs effectively.

ii. Strength

The ClimateAI system significantly reduced the time needed to draft Project Design Documents (PDDs) for Verra climate projects. On average, the AI-generated first draft was completed in minutes compared to the typical multi-hour manual process of researching and writing. Automated extraction and synthesis of user responses streamlined the documentation process, ensuring consistent formatting, standard and methodology information. The AI maintained a standardized structure for PDDs based on Verra guidelines (VCS Standard V4.2). This reduced errors and inconsistencies

commonly found in manually written documents. The system effectively identified and notified users with missing information ensuring completeness of the PDD. The AI was able to answer specific questions about information needed for gathering additional project details.

iii. Limitations and Challenges

- Performance: The AI chat runtime was slow, impacting real-time responsiveness.
- Data Extraction Issues: The system struggled to capture all details from tables, graphs, and equations present in PDDs. These were stored in different text types, so we'd likely need a different tool to deal with this in the future.
- Limited Detail: The AI's outputs were constrained by the information provided by clients, resulting in potential gaps.
- Scalability Challenges: Changing regulations and inactive methodologies required frequent database updates, complicating scalability.
- Domain Expertise: Validating the final output required specialized domain knowledge.
- Validation: Since the team lacks in-depth domain knowledge to validate AI-generated PDD outputs, we plan to seek feedback from our project sponsor or their team, who are more familiar with PDD requirements. Upon their approval, we intend to conduct user testing with selected clients to further validate the outputs and refine the system based on real-world feedback.

Future improvements

In the future, we can continuously update the database with newly approved PDDs, methodologies, guidance, and standards. We built our RAG model on a fixed set of data, but constant updates are necessary going forward. Additionally, we can integrate data from other registries like Gold Standard and PuroEarth. We can also actively track locations in the world with emerging climate related projects to recommend to clients. One of our limitations that we mentioned was with data extraction, where some tables and graphs were stored as a different text type in the pdf files and we were not able to extract them in whole. This impacted the length of our generated PDD by making it shorter than it should have been. In the future, a developer could use a different, specialized tool to convert pdf files to text in order to capture various types of text including tables, graphs, and equations. Finally, we can enhance prompt engineering to generate more detailed and accurate PDDs.

Appendix

a. Link to UI and GitHub
   i. http://34.236.255.34:8501/
   ii. https://github.com/eufouria/verra-crawler
   iii. https://github.com/eufouria/climate-chatbot