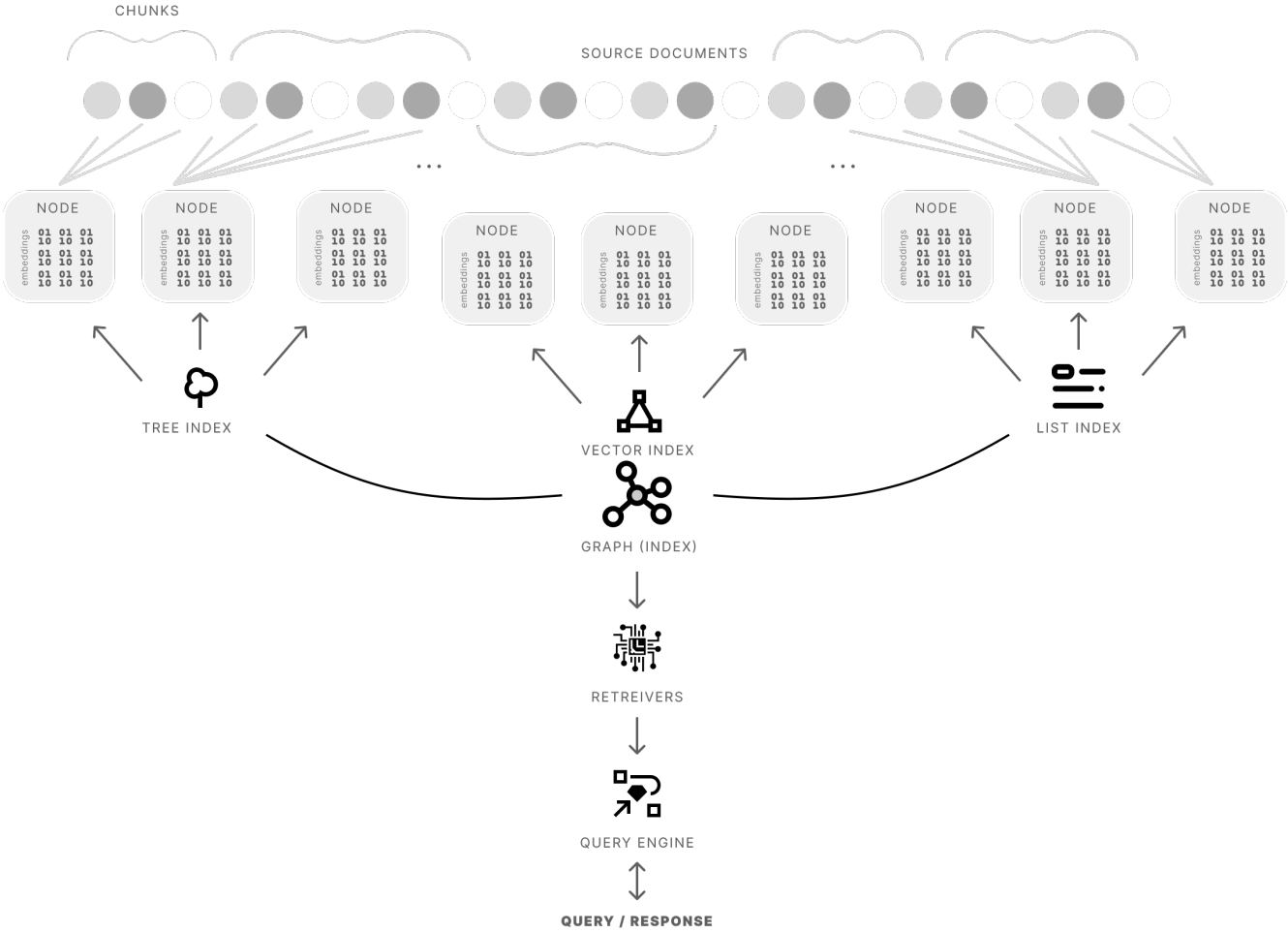
$\mathbb{C}$omposable *graphs* with $\mathcal{MMR}$ queries over *vector* stores.

$\mathbb{C}$omposability allows you to to define lower-level indices for each document, and higher-order indices over a collection of documents. For *e.g.* imagine defining (1) a tree index for the text within each document, and (2) a list index over each tree index (per document) within your collection.[1]



Querying the index or a graph involves a three main components:

Retreivers →	Response Synthesizer →	Query Engine
A retriever class retrieves a set of Nodes from an index given a query.	This class takes in a set of Nodes and synthesizes an answer given a query.	This class takes in a query and returns a Response object. It can make use of Retrievers and Response Synthesizer modules under the hood.

For the query logic itself we will use maximum marginal relevance or $\mathcal{MMR}$. In this we iteratively find documents that are dissimilar to previous results. It has been shown to improve performance for LLM retrievals [2].

The maximum marginal relevance algorithm is as follows:

$$\text{MMR} = \arg \max_{d_i \in D \setminus R} [\lambda \cdot Sim_1(d_i, q) - (1 - \lambda) \cdot \max_{d_j \in R} Sim_2(d_i, d_j)]$$

Here, D is the set of all candidate documents, R is the set of already selected documents, q is the query, Sim_1 is the similarity function between a document and the query, and Sim_2 is the similarity function between two documents. d_i and d_j are documents in D and R respectively.

The parameter λ (mmr_threshold) controls the trade-off between relevance (the first term) and diversity (the second term). If mmr_threshold is close to 1, more emphasis is put on relevance, while a mmr_threshold close to 0 puts more emphasis on diversity.

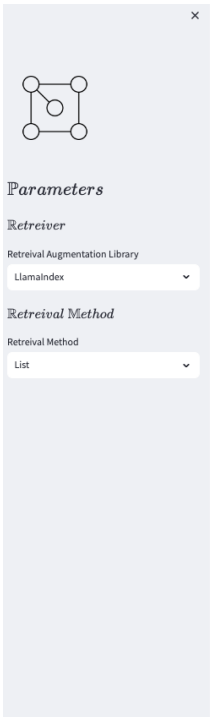
How to use

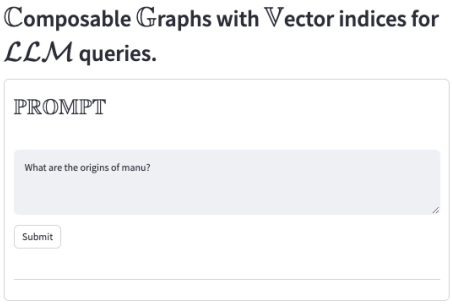
1. Rename `.env.sample` to `.env` and set environment variables `OPENAI_API_KEY`.

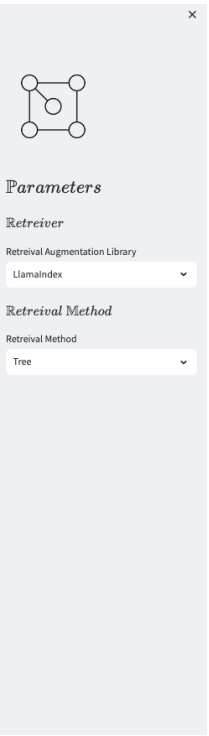
   ```
   # within main folder
   mv .env.example .env
   echo "OPENAI_API_KEY=<your/key/here>" >> .env
   ```

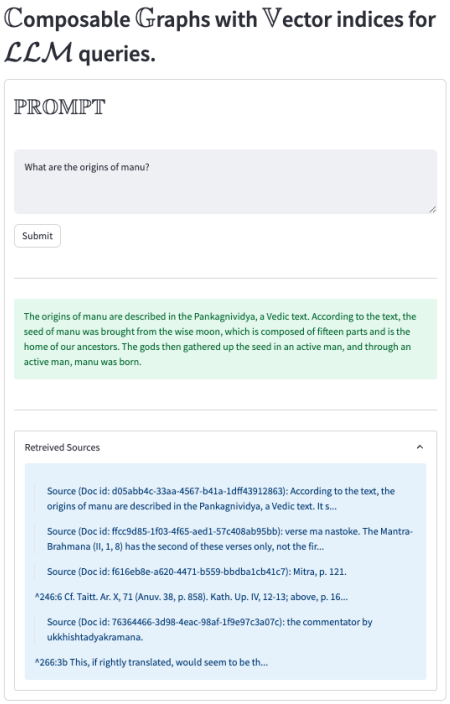
2. Run App

   ```
   streamlit run main.py
   ```









3. Technologies used
    1. openai
    2. llamaindex
    3. langchain
    4. streamlit

Indices

All example indices can be found in the `storage/` folder for loading/reusing in other projects! They are primarily based on CC-commons Indic literature and history corpus from gutenberg.