

KruxAI / ragbuilder

<> Code

Issues 1

Pull requests

Discussions

Actions

Projects

Security

Insights

A toolkit to create optimal Production-ready RAG setup for your data

[docs.ragbuilder.io](#)

Apache-2.0 license

124 stars 13 forks 1 watching 3 Branches 0 Tags Activity Custom properties

Public repository

3 Branches 0 Tags

Go to file t

Go to file

+

Add file

Code

aravind10x

Merge pull request #4 from KruxAI/langchain-upgrade

0b19dfb · 3 days ago

SampleInputFiles	Install.sh & Install.bat. Readme Update	3 weeks ago
SampleSyntheticDataFiles	samplefile update	3 weeks ago
assets	Rename img to assets	3 weeks ago
src	Update executor.py	3 days ago
tests	Install.sh & Install.bat. Readme Update	3 weeks ago
.dockerignore	Updated Readme path in dashboard ...	last week
.env-Sample	Update .env-Sample	4 days ago
Brewfile	Docker fix	2 weeks ago
Dockerfile	Added Docker Compose	4 days ago
LICENSE	Install.sh & Install.bat. Readme Update	3 weeks ago
README.md	Update README.md	4 days ago
docker-compose.yml	Added Docker Compose	4 days ago
install.bat	Updated Install script	3 weeks ago
install.sh	Updated Install script	3 weeks ago
pytest.ini	First Commit	2 months ago
requirements.txt	psycpg- added	4 days ago
setup.py	psycpg- added	4 days ago

README

Apache-2.0 license



# RAGBuilder

RagBuilder is a toolkit that helps you create optimal Production-ready Retrieval-Augmented-Generation (RAG) setup for your data automatically. By performing hyperparameter tuning on various RAG parameters (Eg: chunking strategy: semantic, character etc., chunk size: 1000, 2000 etc.), RagBuilder evaluates these configurations against a test dataset to identify the best-performing setup for your data. Additionally, RagBuilder includes several state-of-the-art, pre-defined RAG templates that have shown strong performance across diverse datasets. So just bring your data, and RagBuilder will generate a production-grade RAG setup in just minutes.

🗂 Ragbuilder\_voiceover\_demo.mp4 ▾

0:00 / 4:28

## Table of Contents

- [Features](#)
- [Installation](#)
- [Set your OpenAI API key](#)
- [Quickstart Guide](#)

## Features

- **Hyperparameter Tuning:** Efficiently identify optimal RAG configurations (combination of granular parameters like chunking strategy, chunking size, embedding models, retriever types etc.) using Bayesian optimization

- **Pre-defined RAG Templates:** Use state-of-the-art templates that have demonstrated strong performance across various datasets.
- **Evaluation Dataset Options:** Choose to generate a synthetic test dataset or provide your own.
- **Automatic Reuse:** Automatically re-use previously generated synthetic test data when applicable.
- **Easy-to-use Interface:** Intuitive UI to guide you through setting up, configuring, and reviewing your RAG configurations.

## Installation

### Option 1: Install using install script:

#### Mac

```
curl -fsSL https://install.ragbuilder.io/mac | bash
```



#### Windows

```
curl -fsSL https://install.ragbuilder.io/win
```



Run Install.bat from command prompt

```
install.bat
```



#### Set your OpenAI API key

Make sure your OpenAI API key is available by setting it as an environment variable. In MacOS and Linux, this is the command:

```
export OPENAI_API_KEY=XXXXX
```



and on Windows it is

```
set OPENAI_API_KEY=XXXXX
```



Now, run ragbuilder on your command line:

```
ragbuilder
```



This will start the Ragbuilder Uvicorn app and open the browser. If the browser window doesn't open automatically, go to <http://localhost:8005/> in your browser to access the RagBuilder dashboard.

### Option 2: Using Prebuilt Docker Image

#### Using Docker Compose

1. Pull the docker-compose.yml file

```
curl -o docker-compose.yml https://raw.githubusercontent.com/KruXAI/ragbuilder/main/docker-compose.yml
```



2. Pull the .env-Sample file to .env File

```
curl -o .env https://raw.githubusercontent.com/KruXAI/ragbuilder/main/.env-Sample
```

3. Edit the .env file to add the necessary keys

4. Start RagBuilder App

```
docker-compose up -d
```

5. Once the services are up, you can access your application via <http://localhost:55003>

## Using Docker Commands

1. Pull docker image from Docker hub

```
docker pull ashwinzyx/ragbuilder:latest
```

2. Run the Docker Container. Create .env as below and use it while running the container. The env file must be in the same directory where the docker command is being run

## Mac

```
docker run -it -v "$(pwd):/ragbuilder" --env-file .env -p 55003:8005  
ashwinzyx/ragbuilder:latest
```

## Windows

```
docker run -d -v %cd%:/ragbuilder --env-file .env -p 55003:8005 ashwinzyx/ragbuilder
```

OR

Provide env variables using command line

```
docker run -p 55003:8005 -e OPENAI_API_KEY=sk-....
```

This will start the Ragbuilder Uvicorn app and open the browser. If the browser window doesn't open automatically, go to <http://localhost:55003/> in your browser to access the RagBuilder dashboard.

*Note: If you are creating your own synthetic dataset for evaluation, save the csv file in the same directory where the docker run command is being executed and provide the file name only*

## Quickstart Guide

Getting started is super easy. To create a new project,

1. Click **New Project** to start building your RAG.
2. **Description:** Describe your use-case. Let's specify "Q&A Chatbot" as the description for our demo.
3. **Source Data:** Specify the path to your source data. This could be a URL, local directory or local file path.  
For the sake of our demo, let's specify the URL: <https://lilianweng.github.io/posts/2023-06-23-agent/>
4. **Select Ragbuilder options:**

- Use Pre-defined RAG Templates - When selected, this'll include pre-defined RAG configuration templates that have demonstrated strong performance across various datasets and related use-cases. These templates will be evaluated against your data, providing you with performance metrics for each pre-defined configuration.
  - Create Custom RAG Configurations - When selected, this'll generate multiple RAG configurations based on detailed parameters like chunking strategy, chunking size, embedding model, retriever type etc. With this option, it is recommended that you opt for the Bayesian optimization option to efficiently identify the near-optimal RAG configuration for your data. More about this in a bit.\*
5. Next, in order to tailor your RAG configurations, you can unselect any specific options you wish to exclude (For eg: Unselecting "Chunking Strategy: Character" will exclude all RAG configurations that have the CharacterTextSplitter). For best results, you may want to leave all settings unchanged. But for our Quickstart demo, we will unselect everything except the below:
- Chunking strategy: Markdown
  - Embedding model: text-embedding-3-large
  - Retriever: Vector DB - Similarity Search
  - Top k: 5
  - LLM: GPT-3.5 Turbo
6. Select optimization approach:
- **Bayesian optimization (Recommended):** Bayesian optimization is a strategy for the optimization of objective functions that are expensive to evaluate. It is particularly useful in scenarios where the function to be optimized is unknown and expensive to compute, such as in hyperparameter tuning for machine learning models or optimizing engineering designs. This is perfect for RAG where we have multiple moving parts, each with multiple parameters
  - **Run all Combinations:** This option runs all possible combinations of the options selected, offering a comprehensive performance analysis of all RAG configurations for your dataset. This option is appropriate if you have selected fewer number of options. Otherwise, this option can be resource intensive as it may yield hundreds or even thousands of unique configurations to compare. *[Note]: This may take several minutes to complete.*
7. Next, in Evaluation dataset options, you have the option to:
- **Use Existing Synthetic Test Data:** If synthetic test data was previously generated for your dataset, this option will appear alongside the path of the existing test data.
  - **Generate Synthetic Test Data from My Dataset:** Create a new synthetic test dataset based on your existing data.
  - **Provide a Manually Created Test Dataset:** Use your own test dataset file (CSV format with "question" and "ground\_truth" columns). For our demo, let's go ahead and create a synthetic test data by selecting the **Generate Synthetic Test Data\*\*** option.
8. Before running the tool, let's review all your selections:
9. Review all the selections and click **Confirm**
10. After processing we should see the dashboard with the results.
11. Click the **View Code snippet** option in the results screen to get the code snippet of the desired RAG configuration. And voila, you've cut down several weeks/months of effort manually creating and evaluating different RAG configuration for your dataset.

## Environment Variables Setup for RagBuilder

This section provides instructions on setting up the environment variables required for the RagBuilder project. These variables need to be configured in a `.env` file located in the same directory where you run the `ragbuilder` command.

## Description

The environment variables are essential for authenticating and configuring various services used by the RagBuilder project. Below are the variables you need to set in your `.env` file.

### Environment Variables

- **OPENAI\_API\_KEY:** The API key for OpenAI services.
- **MISTRAL\_API\_KEY:** The API key for Mistral services.
- **ENABLE\_ANALYTICS:** A boolean flag to enable or disable analytics. Set to `True` or `False`.
- **HUGGINGFACEHUB\_API\_TOKEN:** The API token for HuggingFace Hub.
- **COHERE\_API\_KEY:** The API key for Cohere services.
- **JINA\_API\_KEY:** The API key for Jina services.
- **SINGLESTOREDB\_URL:** The connection string for SingleStoreDB, formatted as `userid:password@host:port/dbname`.
- **PINECONE\_API\_KEY:** The API key for Pinecone services.

### Example .env File

Create a file named `.env` in the directory where you will run the `ragbuilder` command and add the following content, replacing the placeholder values with your actual keys and connection string. Ensure not to use quotes for keys or values

```
# Environment variables for the RagBuilder project
OPENAI_API_KEY=XXXXXX
MISTRAL_API_KEY=XXXXX
ENABLE_ANALYTICS=True
HUGGINGFACEHUB_API_TOKEN=XXXXXX
COHERE_API_KEY=XXXXXX
JINA_API_KEY=XXXXXX
SINGLESTOREDB_URL=userid:password@host:port/dbname
PINECONE_API_KEY=XXXXXX
```



### Instructions

1. Create a new file named `.env` in your project directory.
2. Copy the example content provided above into the `.env` file.
3. Replace the placeholder values with your actual API keys and connection string.
4. Save the `.env` file.
5. Ensure that the `.env` file is located in the same directory where you run the `ragbuilder` command to ensure the environment variables are properly loaded.


### Releases


No releases published

### Packages

No packages published

Contributors 2

 aravind10x

 ashwinzyx Ashwin Aravind

Languages

