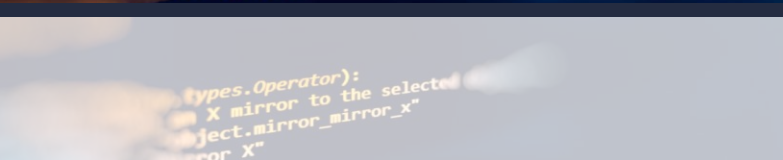


HACKEREARTH RAG APPLICATION – SPRINT 5

Ethan Villalovoz, Molly Iverson, Adam Shtrikman, Chandler Juego



Introduction

Develop a RAG (Retrieval-Augmented Generation) application for HackerEarth that will utilize vector search, knowledge graphs, and a LLM to answer questions and generate content from a knowledge base of more than 10,000 Wikipedia articles.

Sprint Objectives

Large Dataset Testing: Fully embed and process the full Wikipedia dataset to test the RAG model against it.

Vector Search Optimization: Improve vector search results to respond with more relevant information to user queries.

App Packaging and CI/CD: Streamline the build, testing, and deployment process of the app for easier updates.

Custom Use-Cases: Preprocess and embed custom user data to test robustness of the RAG model.

Documentation and Reporting: Continue realigning priorities and goals with client

Feature Implementation

Team Member	Feature	Impact
Ethan	Full Dataset Embedding and Indexing	Expanding to the full Wikipedia dataset enhances the model's ability to retrieve comprehensive, diverse, and well-sourced responses, improving accuracy and contextual relevance for users.
Chandler	Optimize Vector Search Results	Ensures that queries return the most relevant and precise information, reducing noise and ultimately improving response quality. Further tests need to be run.
Molly	App Packaging and CI/CD Pipeline	Allows users to easily run the app on their own machines by providing a consistent, pre-configured environment, eliminating setup complexities and ensuring compatibility across different systems.
Adam	Custom Dataset Integration	Improving the quality of our RAG model against more customized datasets allows for our RAG model to adapt to specific specialized use cases.

Feature Demos

Deployment and Feedback

Deployment: Our client is not using our project, but we've successfully deployed using Docker

Feedback: The client is happy with our progress

Platform and Process: Docker and GitHub Actions CI/CD pipeline. GitHub packages stored on the cloud



Use Testing: Manual testing. CI/CD pipeline only triggers after test pipeline succeeds. Performance is much quicker after deployment

Deployment and Feedback



CI/CD pipeline
using docker-
publish.yml
workflow and
GitHub actions

Packages 2

-  acme10-he-ragapp-backend
-  acme10-he-ragapp-frontend

README MIT license

Running the Application with Docker

If you prefer to use the pre-built Docker images instead of setting up the project manually, follow these steps.

Prerequisites

- Install Docker Desktop: Download and install [Docker Desktop](#).
- Install Docker Compose: Ensure you have `docker-compose` installed. You can install it via Python:

```
pip install docker-compose
```

- LLM API Key: Ensure you have an OpenAI key saved in your environment

```
# For Linux  
export OPEN_API_KEY=value
```

Running the Application

1. Pull the Frontend Image:

```
docker pull ghcr.io/mollyiverson/acme10-he-ragapp-frontend:latest
```

2. Pull the Backend Image:

```
docker pull ghcr.io/mollyiverson/acme10-he-ragapp-backend:latest
```

3. Download the `docker-compose.yml` File:

```
curl -O https://raw.githubusercontent.com/mollyiverson/ACME10-HE-RAGApp/main/docker-compose.yml
```

Versions

1 tagged 13 untagged

latest

Published about 15 hours ago · Digest ...

sha256:37614bd5baead41b005c7fb9c7b127319f3c684

Published 6 days ago

sha256:4b64d21ff8085e05f2070f829e37944a4e88000e

Published 6 days ago

sha256:9f80ff497fd385fe58f397b114cb59b5be798bf5c

Published 7 days ago

Kanban

Overview & Contributions

- **Molly**
 - Packaged and deployed the app using Docker and created a CI/CD pipeline for continuous deployment
 - Wrote project report
- **Ethan**
 - Generated and indexed large embeddings files (~5 GB total) with and without chunking
 - Sprint report
- **Chandler**
 - Improved vector search accuracy
 - Conducted quality assurance and final prep by testing a wide variety of questions and simplifying the UI
- **Adam**
 - Integrated custom dataset (class notes) into embeddings workflow for personalized embedding generation
- **All**
 - Wrote meeting notes for client meetings
 - Edited final report

App packaging/deployment #87

Closed

#89

mollyiverson/ACME10-HE-RAGApp Private



chandyego84 opened 2 weeks ago · edited by chandyego84

Edits · ...

As a developer

I need an efficient and automated packaging and deployment process for the application
So that new versions can be reliably deployed with minimal downtime and effort

Details and Assumptions

- The application needs to be packaged into a deployable artifact (e.g., Docker container, zip file, or other format).
-

Acceptance Criteria

Given a new version of the application code
When the CI/CD pipeline is triggered
Then the application is packaged into a deployable artifact
And the artifact is stored in a versioned repository

Given a successfully packaged application artifact
When the CI/CD pipeline deploys to the staging environment
Then the application is deployed without errors
And automated tests are executed to validate the deployment



Create sub-issue



chandyego84 assigned mollyiverson 2 weeks ago



chandyego84 added this to Capstone Agile Planning 2 weeks ago



chandyego84 moved this to Done in Capstone Agile Planning 2 weeks ago

Assignees

mollyiverson

Labels

documentation

Projects

Capstone Agile

Status Done

Priority

Size

Estimate

Start date

End date

Sprint

Milestone

No milestone

Relationships

None yet

Development

Done 77 / 75 Estimate: 168

This has been completed

ACME10-HE-RAGApp #82

Update Project Report for Sprint 4

5

Sprint 4

ACME10-HE-RAGApp #76

Switch LLM from LLaMA to ChatGPT

3

Sprint 4

ACME10-HE-RAGApp #83

76 switch llm from llama to chatgpt

ACME10-HE-RAGApp #87

App packaging/deployment

Sprint 5

ACME10-HE-RAGApp #89

App Packaging and Deployment

ACME10-HE-RAGApp #91

Revise and Update Project Report

+ Add item

+ Add item

+ Add item

+ Add item

Project Report Refinements

Evidence of Client meetings

Three meetings in Sprint 5:

- Feb 14th
- Feb 21st
- Mar 7th

Evidence of Client meetings

2/14/2025

2. Meeting Summary

Introduction:

- Started with informal chatting and catching up.
- Reviewed current progress and summary of tasks completed from last sprint.

Client's Requirements:

- Client expressed satisfaction with progress and would like to focus on having a demo next week.
- Client suggested we do not focus on deploying the app.

Key Discussion Points:

- **Sprint Priorities:** Reviewed the main deliverables for this sprint, ensuring they align with long-term project goals.

- **Discussed Team Member Tasks To Complete:**

- **Adam:** Expanding dataset creation beyond Wikipedia documents to provide more diverse and useful retrieval data.
- **Chandler:** Research various vector search ranking techniques and implement most suitable one.
- **Molly:** Help with vector search ranking research and implementation.
- **Ethan:** Embedding full wikipedia dataset and uploading instructions to the repo for gaining access to them.

Action Items:

- **Action 1:** Have demoable application with use cases and tests shown. – Due by **02/21/2025**.

Evidence of Client meetings

2/21/2025

2. Meeting Summary

Introduction:

- The meeting began with brief introductions from the project team.
- An overview was provided of the current state of the application.

Client's Requirements:

- Enhance the current custom dataset implementation with advanced features such as graphs and images.
- Due to a git branch issue, the demo was unsuccessful; the client requested to view our Sprint 4 Demo Video as a supplement.

Key Discussion Points:

- **Sprint Priorities:**
 - Reviewed the main deliverables for this sprint, showcasing performance metrics and current implementation speed compared to previous versions.
- **Team Member Contributions:**
 - **Adam:** Discussed and sought feedback on further developing the custom dataset functionality.
 - **Chandler:** Discussed the chunking implementation.
 - **Molly:** Presented performance metrics.
 - **Ethan:** Discussed rebuilding new embeddings containing the full dataset.

Decisions Made:

- **Complete Full Embeddings Generation with the Wikipedia Dataset:**
 - Prioritize this task during the final coding sprint to allow our RAG model to address a wider array of queries.
- **Begin Deployment Work:**
 - Start packaging and deployment efforts during the final coding sprint.
- **Improve Vector Search Ranking:**
 - Implement a ranking mechanism to deliver more relevant responses to users.

Sprint Achievements & Challenges

Achievements:

- **Progress on Large Dataset Implementation:** Initial embedding trials with larger Wikipedia subsets and custom dataset
- **Vector Search Ranking Improvements:** Threshold-based filtering design to ensure high-relevance results and switching to SentenceTransformer Model to enhance semantic search.
- **Packaging & Deployment:** CI/CD pipeline for app build and deployment
- **Custom Data Integration:** embeddings created from custom data

Challenges:

- **Text Chunking Strategy:** test segmentation into larger chunks to better preserve context and improve embedding quality

Next Steps and Sprint Retro

What Went Well:

- Implemented full Wiki dataset embeddings alongside custom data set
- Improved RAG response with quality assurance testing

Next Steps:

- Refine vector search optimization with threshold filtering and expand testing with diverse queries
- Continue to test full dataset with varying embeddings strategies and verify performance
- Poster and client presentation
- Research paper

Conclusion

Key Progress:

- Included full Wikipedia dataset embeddings
- Integrated custom dataset
- Optimized vector search ranking with thresholds and improved sentence embedding model
- Created app deployment pipeline

Thank You!
