BeagleBoard.org
Jul 17, 2025

Table of contents

Chapter 1

Introduction

1.1 Summary links

- **Contributor:** Ketan Thorat
- **Mentors:** Aryan Nanda
- **Code:** **'TBD'_**
- **Documentation:** **'TBD'_**
- **GSoC:** **'TBD'_**

1.2 Status

This project is currently just a proposal.

1.3 Proposal

- Created accounts across OpenBeagle, Discord and Beagle Forum
- The PR Request for Cross Compilation: #200
- Created a project proposal using the proposed template.

Chapter 2

About

- **Forum:** u/ketanthorat.ai (Ketan Thorat)
- **OpenBeagle:** ketanthorat (Ketan Thorat)
- **GitHub:** ketan thorat (Ketan Thorat)
- **School:** K. K. Wagh Institute of Engineering & Research Centre.
- **Country:** India
- **Primary language:** English
- **Typical work hours:** 8AM-5PM Indian Standard Time
- **Previous GSoC participation:** N/A

Chapter 3

Project

Project name: A Conversational AI Assistant for BeagleBoard using RAG and Fine-tuning

3.1 Description

BeagleBoard users often face challenges when troubleshooting errors or getting started as new contributors. This project aims to create an AI-powered assistant to help solve these issues. Using **Retrieval-Augmented Generation (RAG)** and **fine-tuning an open-source language model** (like Llama 3, Mixtral, or Gemma), the chatbot will provide accurate answers, assist with debugging, and guide new contributors through the onboarding process.

3.2 Software

3.2.1 Machine Learning & Model Fine-tuning

- **Hugging Face Transformers** – For model fine-tuning and deployment.
- **PEFT (Parameter Efficient Fine-Tuning)** – For LoRA-based fine-tuning.
- **Hugging Face Datasets** – For handling dataset preprocessing and management.
- **PyTorch** – If using Hugging Face Trainer or other deep learning workflows.
- **TensorFlow/Keras** – If switching from PyTorch to another framework.
- **CUDA (NVIDIA)** – For GPU acceleration during model training.

3.2.2 Data Collection & Preprocessing

- **Gitingest** - For Scraping the docs.beagleboard.io
- **Scrapy** or **BeautifulSoup** – For web scraping documentation and forums.
- **Pandas** – For data structuring and manipulation.
- **Markdown Parsers** – For extracting text from BeagleBoard documentation.

3.2.3 Model Evaluation & Benchmarking

- **BLEU, ROUGE, METEOR** – For evaluating the model performance.
- **Hugging Face Evaluate** – For running model evaluation metrics.

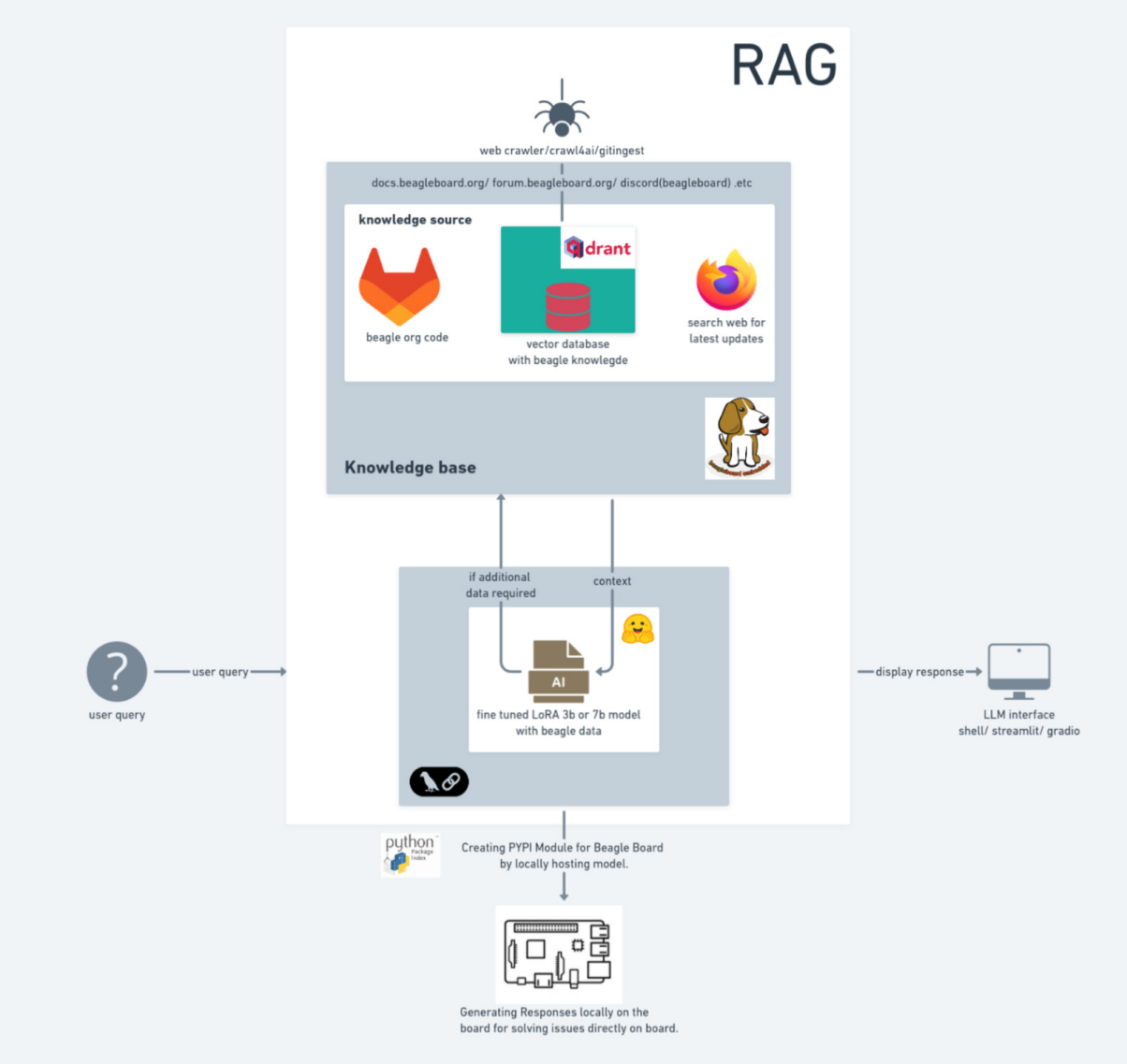

Fig. 1: Architecture of the Implementation of RAG. (This is the basic overview of project, hence doesn't represents the final representation)

3.2.4 Model Deployment & Hosting

- **Hugging Face Hub** – For storing and sharing the trained model.
- **Kaggle Model Hub** – Alternative hosting for public use.
- **TorchServe** or **FastAPI** – For serving inference APIs (if needed).

3.2.5 Interface & User Interaction

- **Gradio** – For building an interactive interface.
- **Streamlit** – Alternative for an easy-to-use web interface.

3.2.6 Retrieval-Augmented Generation (RAG) Implementation

- **FAISS, ChromaDB, or Weaviate** – For vector database storage.
- **LangChain** – For managing RAG workflows.
- **Hugging Face Embeddings** – For converting text into embeddings.

3.2.7 Development & Collaboration

- **OpenBeagle/Github** – For version control.
- **Google Colab** – For quick prototyping.
- **Docker** – If containerizing the model or inference endpoint.

3.3 Hardware

- **BeagleBone AI-64/ BeagleY-AI** - for testing application & performance optimization using quantization techniques.

Chapter 4

Timeline

4.1 Development Timeline

Weeks	Milestones
Community Bonding	Research RAG techniques, finalize tools/models, and decide between CLI vs widget with community feedback.
Week 1	Finalize data sources (Discord, docs, forum), implement markdown chunking, and begin vector DB ingestion.
Week 2	Benchmark retrieval accuracy, refine search parameters, and include reference metadata (URLs, context).
Week 3	Unavailable (Travel). Minimal progress; use buffer time later.
Week 4	Resume work, build RAG generation with LangChain, enable prompt templating and streaming responses.
Week 5	Stream responses via CLI using argparse/rich. If widget chosen, start backend setup and add LLM guardrails.
Week 6	Validate RAG pipeline, gather feedback. Start data prep and formatting for model fine-tuning.
Week 7	Load tokenizer, configure training (lr, batch size), set up CUDA, and begin fine-tuning with Trainer.
Week 8	Complete training, save model (safetensors/pt), evaluate performance, and ensure Hugging Face compatibility.
Week 9	Push model to Hugging Face/Kaggle, expose endpoints, and begin bb-mind PyPI module development.
Week 10	Finalize bb-mind CLI with offline model, wrap inference logic, add system monitoring, mentor feedback loop.
Week 11	Final YouTube video, submit final work, and complete mentor evaluation.

4.2 Timeline summary

Date	Activity
February 27	Connect with possible mentors and request review on first draft
March 4	Complete prerequisites, verify value to community and request review on second draft
March 16	Finalized timeline and request review on final draft
April 6	Submit application
May 8	*Community Bonding Period (May 8th - June 1st)*
June 2	*Introductory YouTube video (June 8th)*
June 9	*Week 2 (June 9th - June 14th)*
June 17	*Week 3 (June 15th - June 21st)*
June 24	*Week 4 (June 22th - June 28th)*
July 1	*Week 5 (June 29th - July 5th)*
July 7	*Week 6 (July 6th - July 12th)*
July 14	*Submit midterm evaluations (July 14th)*
July 21	*Week 7 (July 15th - July 21th)*
July 28	*Week 8 (July 22th - July 28th)*
August 4	*Week 9 (July 29th - Aug 4th)*
August 11	*Week 10 (Aug 5th - Aug 11th)*
August 18	*Final YouTube video and work upload to GSoC site (Aug 18th)*

4.3 Timeline detailed

4.3.1 Community Bonding Period (May 8th - June 1st)

- Research on naive RAG and advanced RAG techniques to improve accuracy of Retrieval in RAG to get accurate answers
- Collaborate and discuss with the community whether a web widget or CLI tool is preferred to get responses from our RAG engine.
- Finalize the tools, framework and model which will be used and get feedback from mentors.
- Discuss the work with mentors and put together precisely the components of my proposal, and include the necessary changes.

4.3.2 Week 1 (June 1st - June 8th)

- Right now some sources for data would be Discord, docs.beagleboard.org, beagle forum and beagle codebase.
- To start ingesting data in the vector database, we first need to decide the chunking strategy for the text corpus.
- Preferably markdown based chunking would be the best strategy for web pages and structured documents.
- Implement chunking with overlap and chunk size and start ingesting the data in the vector database by encoding the data in vectors using a sentence transformer encoder model.

4.3.3 Introductory YouTube video (June 8th)

- The first milestone for the project is an Introductory YouTube Video, where I will present a detailed overview of my GSoC project.
- This video will serve multiple purposes, including introducing the project to the community, explaining the objectives, and outlining the technical roadmap.

4.3.4 Week 2 (June 9th - June 14th)

- Benchmark retrieval accuracy with a dummy dataset on different search params of the vector database to improve retrieval and get accurate chunks to answer the question from the vector database.
- After finalizing the config, retrieve the top 5 documents based on vector search along with some metadata which will be stores at the time of ingestion for eg. URL, prechunk and postchunk.
- Showing the reference URLS i.e sources after answering a question would be a good idea, so users can directly go to source and verify the answer in more depth.

4.3.5 Week 3 (June 15th - June 21st)

- I may be traveling this week due to my personal commitments.
- Meanwhile I will make efforts to stay connected and contribute, but there may be a delay in my responsiveness.

4.3.6 Week 4 (June 22th - June 28th)

- Catchup with the latest updates and changes in the organisation continue working again on Beagle mind.
- Start with writing the generation part of RAG using a framework like langchain by choosing LLM and creating a prompt template to have fields like QUERY, INSTRUCTION and CONTEXT.
- Context field will hold all the chunks retrieved in from the vector database and support the LLM to answer the user query

- Enable streaming of responses in LLM to generate tokens in real time instead showing the complete response at once.

4.3.7 Week 5 (June 29th - July 5th)

- Stream the responses on UI which we decide in the community bonding period.
- For testing we can stream responses on the CLI by using argparse and rich libraries of python that allow to build python cli tools.
- If widget is required start working on it, as it would require hosting the model, vector database and other services on some infrastructure for the web widget to call it and show responses on beagle docs site
- Also add guardrails so the LLM is specific to beagle and its knowledge base.

4.3.8 Week 6 (July 6th - July 12th)

- Test and validate the complete RAG pipeline, by getting feedback from the beagle community on the accuracy of answers from Beagle Mind.
- Start working on finetuning of model
- Preprocess the data into a structured format specific for model finetuning, decide the data format for eg. plain text, key values of instruction and answer.
- Once preprocessed load the data into a hugging face datasets package to handle the dataset.

4.3.9 Submit midterm evaluations (July 14th)

Important: **July 14 - 18:00 UTC:** Midterm evaluation deadline (standard coding period)

4.3.10 Week 7 (July 15th - July 21th)

- Load the tokenizer of the respected model which we have decided to train on.
- Decide configs like learning rate, loss functions, batch size and etc for training the model.
- Start finetuning the model using the hugging face Trainer, this may change if we plan to use the keras or some other framework.
- Install nvidia cuda cores and setup the training environment for CUDA.
- I will also inspect the hardware configuration and fine tune the model. (Right now I have a good graphics card equipped hardware which should be able handle fine tuning locally.)

4.3.11 Week 8 (July 22th - July 28th)

- Complete the training of the model and store the weights locally in safetensors or pt format. This may require few iterations and feedback from the mentors.
- Evaluate the model on some benchmarks like BLEU, ROUGE, METEOR and check response of the fine tuned and original model.
- Save the config and architecture of the model to make it compatible with hugging face transformers.

4.3.12 Week 9 (July 29th - Aug 4th)

- Push the model on hugging face hub or kaggle model based on the decision taken by mentors so it is available for the general public to use and inference the model.
- Also expose the inference endpoints on hugging face hub.
- Carry forward to building PYPI Package for BeagleBoard.

4.3.13 Week 10 (Aug 5th - Aug 11th)

BeagleMind-LLM PyPI Module Development :

- Create Python package (bb-mind) with easy-to-use CLI interface for BeagleBoard.
- Package optimized model files directly within the PyPI module for complete offline usage without API dependencies.
- Build wrapper around optimized model to handle user queries.
- Add system monitoring to display resource usage during inference for testing.
- Get feedback from mentors and make changes as suggested by them.

4.3.14 Final YouTube video and work upload to GSoC site (Aug 18th)

- Upload the final project video to YouTube.
- Submission of final project for mentors evalution.
- Complete the final mentor evaluation.

4.3.15 Final Submission (Aug 25th)

Submit final project video, submit final work to GSoC site and complete final mentor evaluation.

Important: **August 25 - September 1 - 18:00 UTC:** Final week: GSoC contributors submit their final work product and their final mentor evaluation (standard coding period)

September 1 - September 8 - 18:00 UTC: Mentors submit final GSoC contributor evaluations (standard coding period)

4.3.16 Initial results (September 9)

Important: **September 9 - November 9:** GSoC contributors with extended timelines continue coding

November 10 - 18:00 UTC: Final date for all GSoC contributors to submit their final work product and final evaluation

November 17 - 18:00 UTC: Final date for mentors to submit evaluations for GSoC contributor projects with extended deadlines

---

Chapter 5

Experience and approach

- I'm a Robotics Engineering student with experience in embedded systems and IoT projects.
- Since I have a good understanding of hardware, I started exploring hardware-software integration, which led me to contribute to open hardware projects.
- When I first started with BeagleBoard, I found it a bit confusing. I want to contribute to this project to help make it easier for others to get started. Along with that, I have contributed to documentation fixes and community engagement to make resources more accessible.
- With my background in Python, AI agents, and Embedded Linux from my robotics studies, I believe I'm a great fit for this project.
- As a community manager in ML Nashik, backed by Google Developers, I have taken workshops on AI Agents, AI Algorithms, RAG developments, and the fusion of hardware & software.

Contributions to openbeagle.org/docs:

- Resolved BrokenProduct Link issue: #172 (merged)
- Changed bb-imager link: #174 (merged)

5.1 Contingency

If I face any blockers while my mentor is unavailable, I will take the following approach:

1. The BeagleBoard community is highly active and resourceful, making it an excellent place to seek guidance and clarifications.
2. I will explore the official BeagleBoard.org documentation and consult relevant technical forums to find potential solutions.
3. Before escalating an issue, I will methodically debug errors, analyze logs, and experiment with alternative solutions to resolve the problem independently.

5.1.1 Benefits to the Community After Project Completion

The AI assistant will provide the following advantages:

- **Faster Responses to Beagle-Specific Questions** It will quickly address queries related to BeagleBoard, improving user experience and troubleshooting efficiency.
- **Preservation of Community Knowledge** By integrating with documentation and forums, the assistant will help retain and organize valuable information for future users.
- **Affordable** Since it uses LoRA, the assistant will be cost-effective to run and can easily handle more users as the BeagleBoard community grows.

- **Offline Capability** By running locally on BeagleY-AI, the assistant works without internet connectivity, making it useful in remote or restricted environments.
- **Privacy-Preserving** All data processing occurs on the local device, ensuring user queries and information remain private and secure.

5.2 Misc

I will adhere to all GSoC general requirements and submit my merge request to the BeagleBoard GitHub repository. The link to the merge request will be shared after finalizing my initial implementation.

5.3 References

1. Hugging Face Transformers
2. ChromaDB Documentation
3. BeagleBoard Documentation
4. PEFT Fine-tuning