final rag system demo

final submission- demo and writeup

Welcome to the final project page. Retrieval Augmented Generation (RAG) is a framework for LLM powered systems that make use of external data sources. By doing so, RAG overcomes knowledge cutoff issues by leveraging external data sources to update the LLM's understanding of the world.

You will be executing a demonstration for your project orchestrates with LLMs and produces a finish project: a Retrieval Augmented Generative (RAG) system with data of your choosing. With your LLMs, you will pair it with common natural language processing techniques to retrieve information from a database. This laboratory may be helpful for getting your server up and running. It is likely that we will need to ensure that we have sufficient GPU resources. Refer to the earlier labs in the class.

demo project parameters

RAG serves as an alternative to re-training or fine-training models, which is expensive and difficult, and would often require repeated re-training to regularly update the model with new information. Recency is one known knowledge cutoff issue and relate to several applications (e.g., news outlets). Another knowledge cutoff would be exposure to specific or esoteric domain knowledge subject matter (e.g., medical documents, documents in particular scientific fields). It is a more flexible and less expensive way to overcome knowledge cutoffs, and is simple to implement as it pertains mostly to inference.

Feel free to pick any topic that you have data for. You can form groups of up to four but *only submit one for the entire group.* In your writeup, have a contributions sections denoting what each member worked on. Your solution needs to be entirely produced by code that you have written and models that **you** are serving.

- Require a front end (e.g., through streamlit)
- Provide a database that is no less than 10k entries
- LLMs are entirely local (i.e., on GCP or metal) / native
- Provide clickable citation to the data source (article and passage)

You may store your data in any backend (e.g., relational databases like SQL structures, knowledge graph, vector databases, key/value stores, etc.) You may orchestrate with any software (e.g., Airflow, Haystack, Ollama, Langchain, etc.) or implement your own. Your write-up must include a systems diagram of your system.

submission instructions

Commit all your code to your repository, and submit via Gradescope – one submission per team – along with the following artifacts.

- Project Writeup (PDF Format)

    - Click on `project_template.tex` to edit your project
    - Review the Heilmeier Catechism
- Slides Linked to Master Deck
    - Link your slide deck to the master presentation slide deck on slide 2
    - Your elevator pitch for your work should be *at most* three minutes. We will be strict on timelines.
- Project Repository (Github Repository via Link)
    - Include a README.md on how to setup and run the project. This should be straightforward.
    - Containerize your solution with Docker so we don't need specific libraries or software.
- Demonstration (DNS unnecessary)
    - This will need to be up on presentation day. Ensure that we can access your server (which can be an IP address)

---

rubric and criterion

There are two components your project: its presentation and your technical delivery. This delivery will come in the form of a real-time demonstration of your engineering. In order to receive credit, your **must have a real-time inference** component. The delineation between the two contributions will be graded on the following.

40% : **Documentation** : Written and oral delivery of project rationale and justification of approach

- Project Writeup - PDF template (30%)
    * Motivation and Impact: A future iteration of this project can make a difference
    * Background and Related Work: Project is well-researched within and beyond course scope
    * Modeling Methodology: Robustness to pitfalls like class imbalance and overfitting
    * Evaluation and Analysis: Justification of the approach is sound
- Oral Delivery Slide Deck (10%)

60% : **Technical** : Demonstration of your work, handling of corner cases, technical correctness

- *Accessibilty and Performance*: The endpoint is open and is accessible to the public. Results arrive in a reasonble time, and path to additional scaling is apparent
- *Problem Complexity and Data Scale*: Data needs to be meaningfully challenging and unique. There are oftentimes at least 10k rows of meaningful amount of data
- *Accuracy (Quality)*: Do we have confidence that the LLM will not hallucinate?
- *Code / Data Provenance*: Algorithms must be replicable and containerized. Ensure your RAG introduces
    * Reproducibility: If someone wanted to, couuld they re-create your capability? Appropriately cite your dataset, which needn't be open source.
    * Verifiability: Can we follow citations to the appropriate article

---

presentation and grading

During the final presentation day, we will be demonstrating our capabilities and projects.

- **4:00pm** - Lightning papers for those working on publications. Target 5 minutes each presentation and allocate 5 minutes for question and answering.
- **4:15pm** - Each team will have five minutes to introduce their project in presentation. Prior or during this time, teams should start their servers so that the demonstration can commence. Instructor and TA's will operate the end-point in real-time from their laptops.

- **4:45pm** - Instructional staff issues a series of queries designed to verify the main value proposition, corner cases, and latency assessment. Please have preset queries and evidence that your project is functioning within your own expectations. For example, for RAG system queries, identify the passage and provide evidence that the LLM has retrieved from this passage.
- **5:45pm** - Students query each others' RAG and PAL systems. If demonstrations have faced errors, debugged capabilities can be re-graded.
- **6:15pm** - Wind down and farewell