DeepQuery: An Arena-based Retrieval Augmented Generation Evaluation Platform

Taaha Khan, Akash Ravandhu, Luke Merletti, Caleb Li, Sayam Goyal
Department of Computer Science, Purdue University

Introduction

When businesses are considering implementing a Retrieval-Augmented Generation (RAG) system, they often face the same question of tradeoffs: what is the best balance of accuracy for their specific application, operational costs, and system latency? For large language models, platforms like LMArena and ArtificialAnalysis provide incredible benchmarking that helps inform businesses and users on which AI to use.

We wanted to build a reliable and scalable benchmarking methodology to test RAG systems against each other using a variety of factors like base accuracy, user preference of responses, latency, price, and several other available metrics, so we started the DeepQuery project. In DeepQuery, we've set up pipelines for industry RAG approaches, and partnered with Captain to benchmark and compare their cutting-edge systems. We tackle the question of ranking by testing across a variety of datasets, and hope to come to a stronger understanding of the best retrieval agent methodologies and performances.

Pipelines

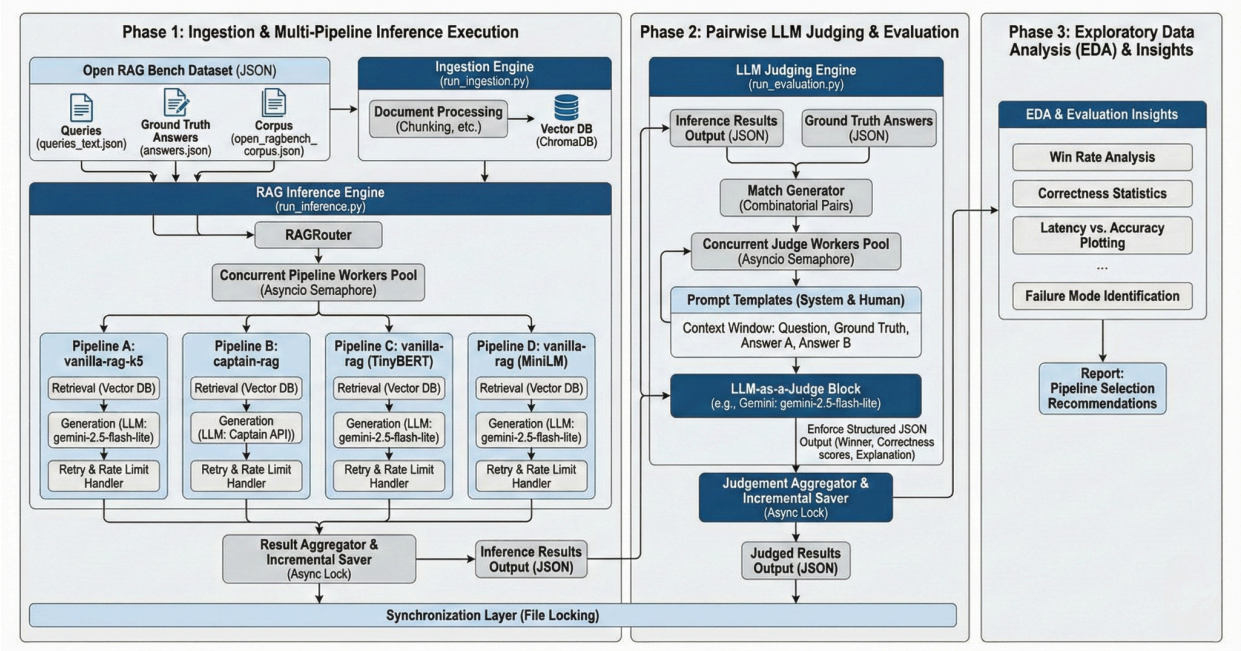
Several custom implementations of RAG agents are established as opponents, along with Captain's search functionality. Broadly, 6000 character chunks with 200 character overlaps were used, with gemini-embedding-001 as the embedding model and Gemini-2.5-Flash-Lite for the generation model under varying retrieval configurations.

Evaluation

To evaluate retrieval quality, we use an LLM-as-a-Judge framework that performs grounded pairwise comparisons across pipelines. For each question, two pipeline answers are compared against the dataset's ground-truth answer. The judge then issues a correctness evaluation with respect to a ground truth label.

We use the OpenRAG-Bench text dataset for our initial experiments: https://huggingface.co/datasets/vectara/open_ragbench. It contains a diverse corpus of papers across a variety of Arxiv categories, with extractive, abstractive, text-based, and table-based question-answer pairs from throughout the context.

Methodology



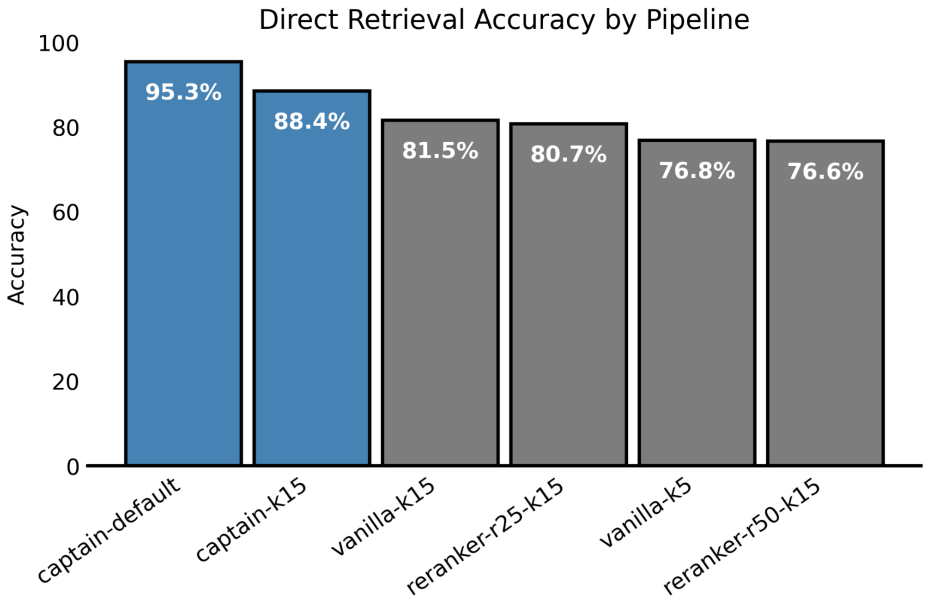
This evaluation methodology follows a structured pipeline designed to quantify RAG agent performance by moving from raw data ingestion to sophisticated relative ranking. It begins with an ingestion and inference phase, where a source corpus is partitioned into searchable documents across multiple pipeline configurations (such as vanilla RAG vs. reranker-based models). An asynchronous router then executes a large-scale inference run, collecting responses, metadata, and latencies from each agent for a shared set of queries, ensuring a consistent baseline for comparison.

The core of the evaluation relies on a structured LLM-based judging process where each RAG agent's response is judged independently. Using a model like Gemini-2.5-Flash as a Judge, the system performs a binary factual check against ground-truth answers. This step is isolated; the judge does not see the responses of other agents at this stage, focusing solely on whether a specific agent provided a factually accurate answer and documenting the reasoning behind that decision.

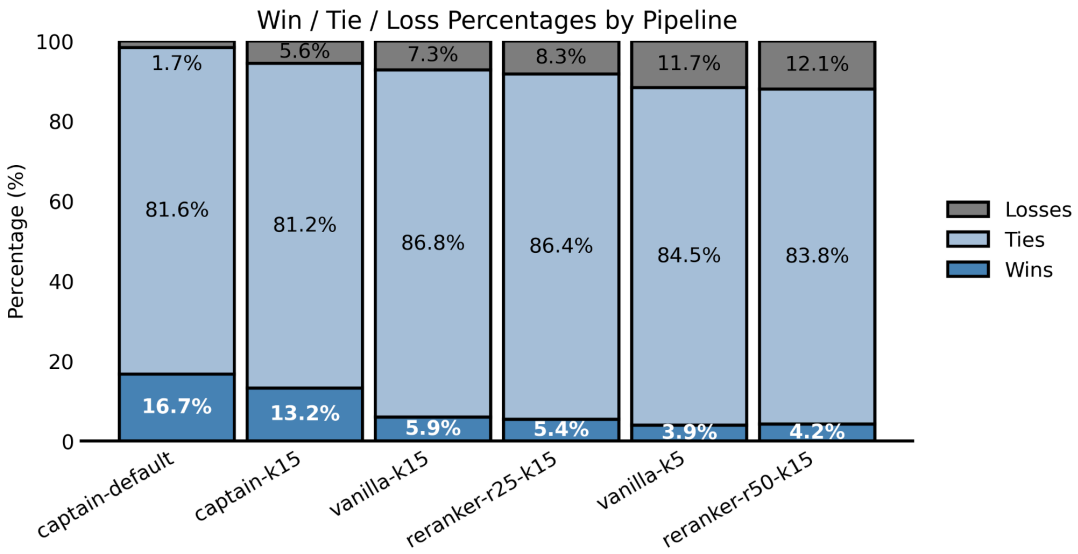
Finally, these independent scores are used to construct pairwise match combinations to determine relative agent hierarchy. By comparing the correctness results of two

pipelines for the same query, the system records wins, losses, or ties (e.g., a "tie-correct" occurs when both agents are accurate). These head-to-head results are then fed into a Bradley-Terry model, a probabilistic approach that converts pairwise win-loss data into a single, calibrated performance score for each agent. This allows for a robust ranking that reflects how much more likely one RAG configuration is to succeed compared to another across the entire dataset.

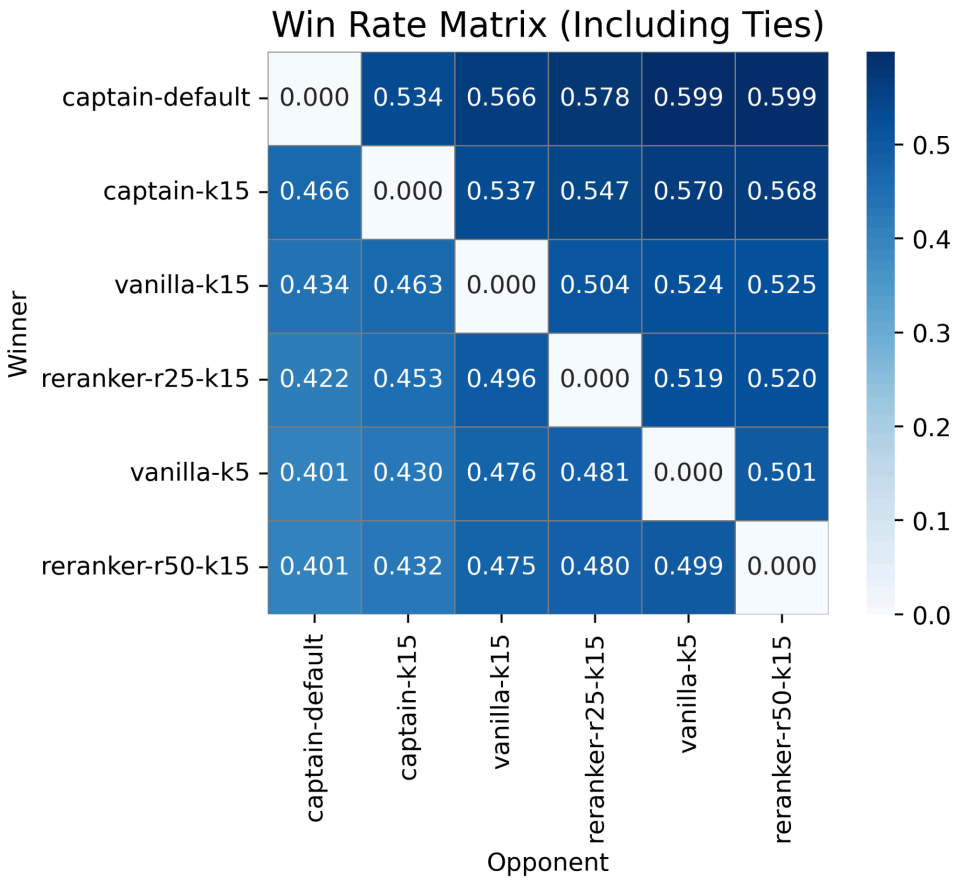
Results



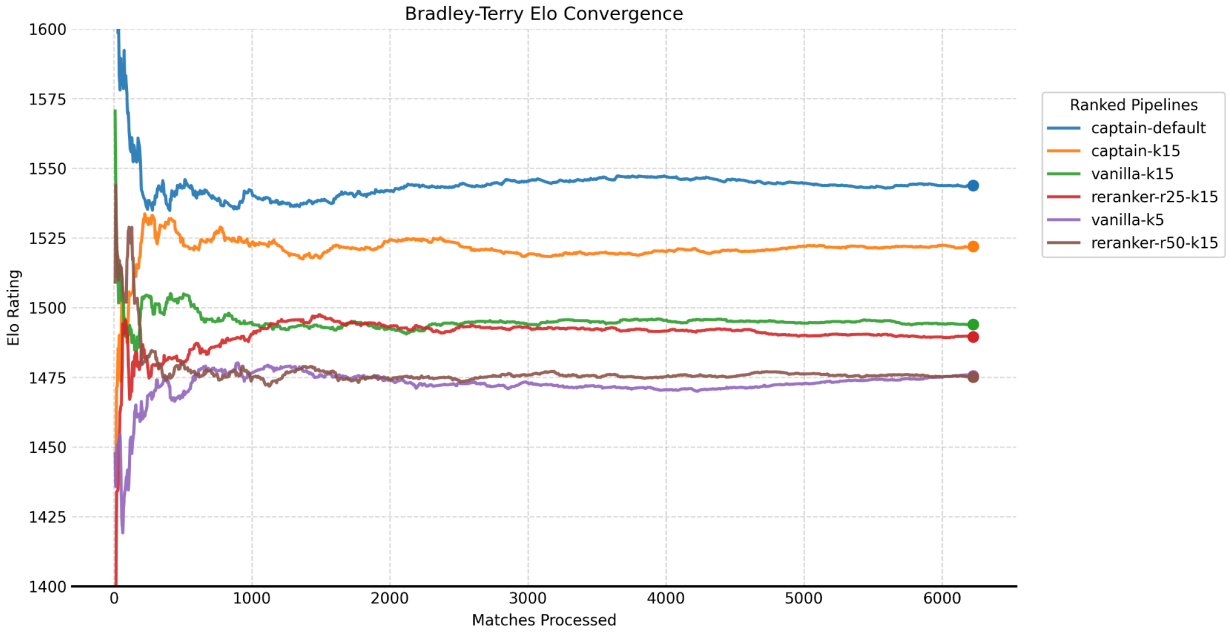
Direct retrieval results are determined by exclusively checking the pipeline's LLM generated answer with respect to the query and ground truth label in the dataset. A standardized LLM-as-a-Judge is utilized for this determination, providing its reasoning and evaluation for each unique query independently for every generation.



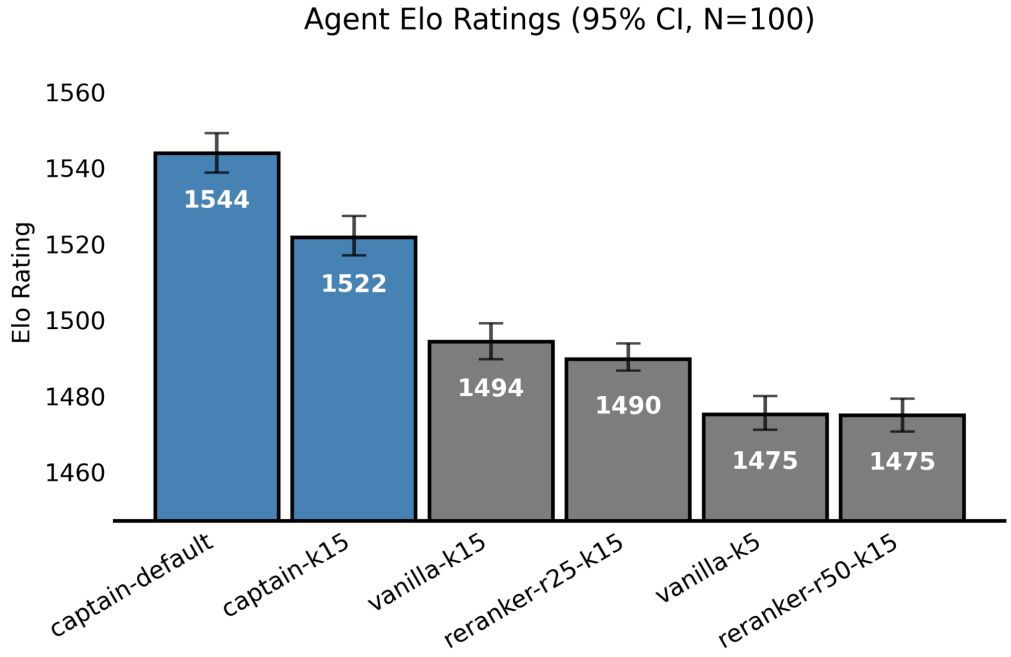
All combinations of head-to-head matchups are generated for the same query and each different retrieval agent, and win percentages are computed based on a strict, rule-based metric to eliminate style and length biases in relative scoring, where wins or losses are only issued if one agent is deemed correct and the other is incorrect by the independent judges, and remaining responses are tied.



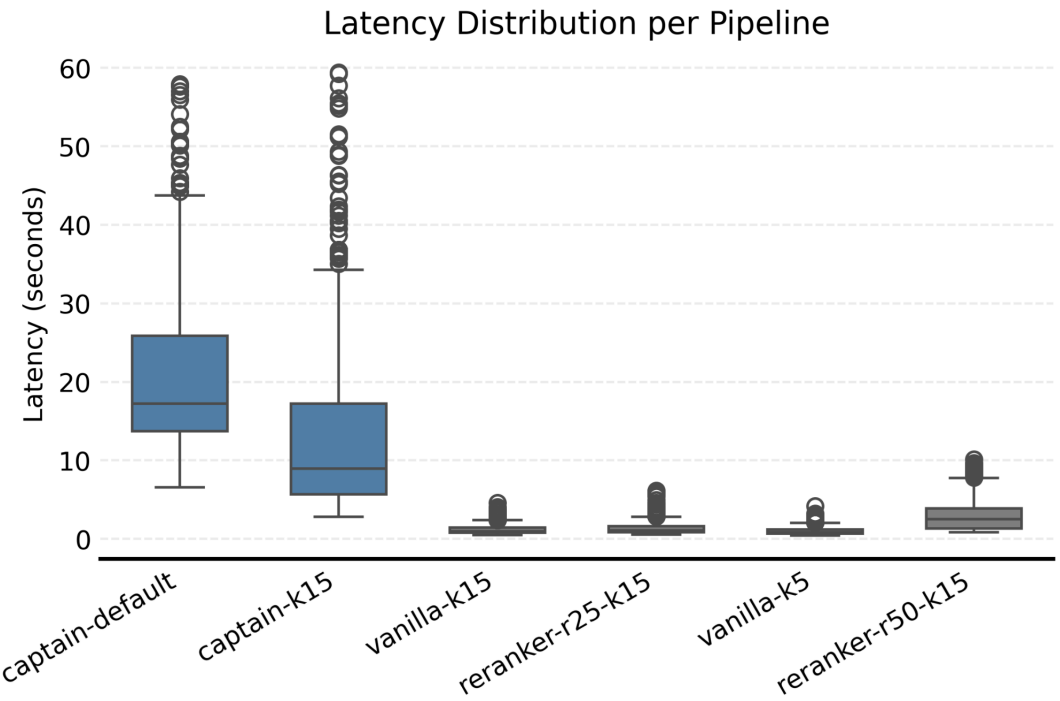
Head-to-head matchup results are also computed to determine the win rates for each agent against each other for a relative calculation, where ties are counted as half wins for both agents.



We fit a Bradley-Terry regression to every combination of individual matchups for the unique queries. The model indicates the relative strength of each system using the strict correctness preference.



Using LMArena-style bootstrapping methods, a confidence interval on the established ratings can be determined to show the potential distribution possible in scores.



We also evaluated the general latency of retrieving the responses for each pipeline. Each of the timings was measured from initial query submission to final token response output, including LLM generation, retrieval, summarization, and API transmission.

Future Steps

DeepQuery is still a work in progress, so we are currently working on bringing a few new key features to the service.

- **More RAG Architectures:** We are working on providing many different RAG configurations and architectures to provide a thorough evaluation suite.

- **Bring Your Own Dataset:** We are working on providing support for uploading your own internal dataset to test each RAG system against. This way, you can understand which application is best for your specific use case.